INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 02/09826

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L12/56 H04L12/18

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, INSPEC, COMPENDEX

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X Y	US 6 108 706 A (SARGENT JAMES RANDALL ET AL) 22 August 2000 (2000-08-22) the whole document ---	1-4,6-9, 11-13 5,10,14
X	EP 1 024 661 A (HUGHES ELECTRONICS CORP) 2 August 2000 (2000-08-02) the whole document ---	1-14
Y	US 6 032 197 A (MORAN BRIAN ET AL) 29 February 2000 (2000-02-29) the whole document ---	5,10,14
A	US 6 081 907 A (SARGENT JAMES RANDALL ET AL) 27 June 2000 (2000-06-27) the whole document --- -/--	1-14

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
2 December 2002	09/12/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Mannekens, J

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 02/09826

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	WO 00 79734 A (UNIV COLUMBIA) 28 December 2000 (2000-12-28) the whole document -----	1-14

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 02/09826

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 6108706 A	22-08-2000	NONE	
EP 1024661 A	02-08-2000	EP 1024661 A2	02-08-2000
US 6032197 A	29-02-2000	NONE	
US 6081907 A	27-06-2000	NONE	
WO 0079734 A	28-12-2000	AU 5879800 A	09-01-2001
		WO 0079734 A1	28-12-2000

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 02/09833

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L29/06 H04L12/18 H04Q7/22 H04L12/28

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ, INSPEC, COMPENDEX, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 1 024 661 A (HUGHES ELECTRONICS CORP) 2 August 2000 (2000-08-02) page 6, column 10, line 52 -page 7, column 11, line 18 page 8, column 13, line 49 -column 14, line 12 page 12, column 21, line 9 -page 15, column 28, line 11 page 16, column 29, line 25 -page 18, column 33, line 35 ---	1-16, 18-27, 29-44, 46-51
A	US 6 032 197 A (MORAN BRIAN ET AL) 29 February 2000 (2000-02-29) column 1, line 26 - line 35 column 4, line 7 - line 53 column 6, line 59 -column 9, line 31 --- -/--	8

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
18 November 2002	26/11/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Karavassilis, N

1

In ional Application No
PCT/US 02/09833

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	WO 00 36804 A (RUDKIN STEVEN ;SAROM ING (GB); BELL SARAH (GB); BRITISH TELECOMM ()) 22 June 2000 (2000-06-22) abstract page 7, line 25 -page 8, line 9 page 12, line 12 - line 16	15,26,43
A	EP 1 075 118 A (NORTEL NETWORKS LTD) 7 February 2001 (2001-02-07) abstract page 3, column 3, line 39 - line 46	33
A	HANDLEY M ET AL: "SDP: Session Description Protocol" RFC 2327, April 1998 (1998-04), XP002101463 page 1, paragraph 1 -page 2, paragraph 2 page 3, paragraph 5	1-51
A	WO 00 33535 A (FAIRMAN IAN RALPH ;SMITH ALAN PHILIP (GB); RUDKIN STEVEN (GB); SAR) 8 June 2000 (2000-06-08) page 4, line 19 -page 5, line 17 page 6, line 11 -page 7, line 15 page 13, line 14 - line 25 page 28, line 5 - line 26 page 35, line 1 - line 25	1-51

1

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 02/09833

Patent document cited in search report		Publication date	Patent family member(s)			Publication date
EP 1024661	A	02-08-2000	EP	1024661	A2	02-08-2000
US 6032197	A	29-02-2000	NONE			
WO 0036804	A	22-06-2000	AU	1286700	A	19-06-2000
			AU	1395900	A	03-07-2000
			EP	1133861	A1	19-09-2001
			EP	1142267	A1	10-10-2001
			WO	0036804	A1	22-06-2000
			WO	0033535	A1	08-06-2000
			JP	2002532012	T	24-09-2002
			JP	2002533023	T	02-10-2002
EP 1075118	A	07-02-2001	EP	1075118	A2	07-02-2001
WO 0033535	A	08-06-2000	AU	1286700	A	19-06-2000
			AU	1395800	A	19-06-2000
			EP	1133861	A1	19-09-2001
			EP	1131935	A1	12-09-2001
			WO	0033534	A1	08-06-2000
			WO	0033535	A1	08-06-2000
			JP	2002532011	T	24-09-2002
			JP	2002532012	T	24-09-2002
			AU	1395900	A	03-07-2000
			EP	1142267	A1	10-10-2001
			WO	0036804	A1	22-06-2000
			JP	2002533023	T	02-10-2002

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 02/09832

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L29/06 H04L12/18 H04Q7/22 H04L12/28

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, WPI Data, PAJ, INSPEC, COMPENDEX, IBM-TDB

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 1 024 661 A (HUGHES ELECTRONICS CORP) 2 August 2000 (2000-08-02) page 6, column 10, line 52 -page 7, column 11, line 18 page 8, column 13, line 49 -column 14, line 12 page 12, column 21, line 9 -page 15, column 28, line 11 page 16, column 29, line 25 -page 18, column 33, line 35 ---	1-11
A	US 6 032 197 A (MORAN BRIAN ET AL) 29 February 2000 (2000-02-29) column 1, line 26 - line 35 column 4, line 7 - line 53 column 6, line 59 -column 9, line 31 --- -/--	8

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
18 November 2002	26/11/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Karavassilis, N

International Application No
PCT/US 02/09832

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	HANDLEY M ET AL: "SDP: Session Description Protocol" RFC 2327, April 1998 (1998-04), XP002101463 page 1, paragraph 1 -page 2, paragraph 2 page 3, paragraph 5	1-11

Form PCT/ISA/210 (continuation of second sheet) (July 1992)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 02/09832

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
EP 1024661	A	02-08-2000	EP	1024661 A2	02-08-2000
US 6032197	A	29-02-2000	NONE		

Form PCT/ISA/210 (patent family annex) (July 1992)

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 02/09834

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L12/18 H04L29/06 H04L12/56

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 1 024 661 A (HUGHES ELECTRONICS CORP) 2 August 2000 (2000-08-02) abstract; figure 1 column 1, paragraph 1 column 5, paragraph 14 -column 6, paragraph 16 column 11, paragraph 30 -column 14, paragraph 36 column 20, paragraph 57; figure 7 column 29, paragraph 84 column 31, paragraph 90 ---	1-10
X	WO 00 36804 A (RUDKIN STEVEN ;SAROM ING (GB); BELL SARAH (GB); BRITISH TELECOMM () 22 June 2000 (2000-06-22) page 1, line 1 -page 2, line 6 page 11, line 9 -page 15, line 13 --- -/--	1-10

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
7 February 2003	27/02/2003

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Hultsch, W

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	US 6 032 197 A (MORAN BRIAN ET AL) 29 February 2000 (2000-02-29) abstract; figure 2 ---	10
A	HANDLEY M ET AL: "SDP: Session Description Protocol" IETF RFC 2327, April 1998 (1998-04), pages 1-40, XP002101463 the whole document ---	1-9
A	BURMEISTER C ET AL: "ROBUST HEADER COMPRESSION (ROHC) " INTERNET DRAFT, XX, XX, December 2000 (2000-12), pages 1-122, XP002901751 page 4 -page 5 ---	10
A	"Universal Mobile Telecommunications System (UMTS); Radio Interface for Broadcast/Multicast Services (3GPP TR 25.925 version 3.3.0 Release 1999)" ETSI TR 125 925 V3.3.0, 1 December 2000 (2000-12-01), pages 1-35, XP002230388 page 17 -page 18; figure 6.5 -----	4

INTERNATIONAL SEARCH REPORT

PCT/US 02/09834

Patent document cited in search report		Publication date	Patent family member(s)			Publication date
EP 1024661	A	02-08-2000	EP	1024661	A2	02-08-2000
WO 0036804	A	22-06-2000	AU	1286700	A	19-06-2000
			AU	1395900	A	03-07-2000
			EP	1133861	A1	19-09-2001
			EP	1142267	A1	10-10-2001
			WO	0036804	A1	22-06-2000
			WO	0033535	A1	08-06-2000
			JP	2002532012	T	24-09-2002
			JP	2002533023	T	02-10-2002
US 6032197	A	29-02-2000	NONE			

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 03/26897

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 G06F1/00 H04L9/08

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 G06F H04L

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 1 213 943 A (LUCENT TECHNOLOGIES INC) 12 June 2002 (2002-06-12) abstract column 3, line 19 - line 28	1-8
Y	MENEZES A J ET AL: "KEY LAYERING AND CRYPTOPERIODS, PASSAGE" 1997 , HANDBOOK OF APPLIED CRYPTOGRAPHY, CRC PRESS SERIES ON DISCRETE MATHEMATICES AND ITS APPLICATIONS, BOCA RATON, FL, CRC PRESS, US, PAGE(S) 551-553,577-581 XP002202082 ISBN: 0-8493-8523-7 page 551 -page 553 page 577 -page 581	1-8

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
7 January 2004	16/01/2004

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	San-Bento Furtado, P

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	WO 00 02406 A (NOKIA NETWORKS OY ;EKBERG JAN ERIK (FI)) 13 January 2000 (2000-01-13) abstract page 8, line 13 -page 9, line 2 page 10, line 10 -page 14, line 1	1-8
Y	BERKOVITS S: "How to Broadcast a Secret" ADVANCES IN CRYPTOLOGY- EUROCRYPT. INTERNATIONAL CONFERENCE ON THE THEORY AND APPLICATION OF CRYPTOGRAPHIC TECHNIQUES, SPRINGER VERLAG, DE, 11 April 1991 (1991-04-11), pages 535-541, XP002202083 page 535, paragraph 3 -page 536, paragraph 2	2-8
X	WO 02 061572 A (ITAGAKI TAKATOSHI ;MORIGUCHI ATSUSHI (JP); NATSUNO TAKESHI (JP); N) 8 August 2002 (2002-08-08)	1
Y	abstract paragraph '0024! paragraph '0051! - paragraph '0058! -& EP 1 248 188 A 9 October 2002 (2002-10-09)	2-8
P,X	WO 02 080449 A (QUALCOMM INC) 10 October 2002 (2002-10-10) the whole document	1-8
P,X	WO 03 032573 A (QUALCOMM INC) 17 April 2003 (2003-04-17) the whole document	1-8

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US 03/26897

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
EP 1213943 A	12-06-2002	US 2002071558 A1	13-06-2002
		EP 1213943 A1	12-06-2002
		JP 2002232418 A	16-08-2002
WO 0002406 A	13-01-2000	FI 981565 A	08-01-2000
		AU 4912199 A	24-01-2000
		DE 19983405 T0	31-05-2001
		WO 0002406 A2	13-01-2000
		GB 2355157 A ,B	11-04-2001
		JP 2002520923 T	09-07-2002
WO 02061572 A	08-08-2002	BR 0202291 A	10-06-2003
		CA 2397711 A1	08-08-2002
		CN 1455894 T	12-11-2003
		EP 1248188 A1	09-10-2002
		WO 02061572 A1	08-08-2002
		NO 20024672 A	29-11-2002
		US 2002194474 A1	19-12-2002
WO 02080449 A	10-10-2002	US 2002141591 A1	03-10-2002
		EP 1374477 A1	02-01-2004
		EP 1374528 A2	02-01-2004
		EP 1374483 A2	02-01-2004
		EP 1374529 A2	02-01-2004
		EP 1374440 A2	02-01-2004
		EP 1374506 A2	02-01-2004
		EP 1378145 A1	07-01-2004
		WO 02080401 A2	10-10-2002
		WO 02080588 A2	10-10-2002
		WO 02080609 A1	10-10-2002
		WO 02080488 A2	10-10-2002
		WO 02080489 A2	10-10-2002
		WO 02080490 A2	10-10-2002
		WO 02080589 A2	10-10-2002
		WO 02080590 A2	10-10-2002
		WO 02080454 A2	10-10-2002
		WO 02080449 A1	10-10-2002
		US 2002141365 A1	03-10-2002
		US 2002181423 A1	05-12-2002
		US 2003134655 A1	17-07-2003
		US 2002141371 A1	03-10-2002
		US 2002142730 A1	03-10-2002
		US 2002142757 A1	03-10-2002
		US 2003228861 A1	11-12-2003
		US 2002141391 A1	03-10-2002
		US 2002141447 A1	03-10-2002
		US 2003039361 A1	27-02-2003
WO 03032573 A	17-04-2003	US 2003070092 A1	10-04-2003
		WO 03032573 A2	17-04-2003

INTERNATIONAL SEARCH REPORT

PCT/US 02/09835

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L9/08 H04Q7/38

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L H04Q

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

EPO-Internal, INSPEC, WPI Data, PAJ

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	MENEZES, OORSCHOT, VANSTONE: "Handbook of Applied Cryptography" CRC PRESS SERIES ON DISCRETE MATHEMATICES AND ITS APPLICATIONS, BOCA RATON, FL, CRC PRESS, US, 1997, pages 551-553,577-581, XP002202082 ISBN: 0-8493-8523-7 page 551 -page 553 page 577 -page 581 ---	1-24
A	BERKOVITS S : "How to Broadcast a Secret" ADVANCES IN CRYPTOLOGY - EUROCRYPT '91 CONFERENCE. SPRINGER-VERLAG, 11 April 1991 (1991-04-11), pages 535-541, XP002202083 Brighton, UK, ISBN: 3-540-54620-0 page 535 -page 536 ---	1-24
	-/--	

[X] Further documents are listed in the continuation of box C.

[ ] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
13 June 2002	08/07/2002

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31–70) 340–2040, Tx. 31 651 epo nl, Fax: (+31–70) 340–3016	Carnerero Álvaro, F

Form PCT/ISA/210 (second sheet) (July 1992)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	MARCHENT B G ET AL: "Intelligent control of mobile multimedia systems" VEHICULAR TECHNOLOGY CONFERENCE, 1998. VTC 98. 48TH IEEE OTTAWA, ONT., CANADA 18-21 MAY 1998, NEW YORK, NY, USA,IEEE, US, 18 May 1998 (1998-05-18), pages 2047-2051, XP010288261 ISBN: 0-7803-4320-4 the whole document -----	5,6,21

PATENT COOPERATION TREATY

PCT

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

(PCT Article 36 and Rule 70)

Applicant's or agent's file reference 010498WO	**FOR FURTHER ACTION** See Notification of Transmittal of International Preliminary Examination Report (Form PCT/IPEA/416)	
International application No. PCT/US02/09829	International filing date *(day/month/year)* 28 March 2002 (28.03.2002)	Priority date *(day/month/year)* 28 March 2001 (28.03.2001)
International Patent Classification (IPC) or national classification and IPC IPC(7): H04 L 12/18 and US Cl.: 370/328,340,389,392,469,470,474,476		
Applicant QUALCOMM INCORPORATION		

1. This international preliminary examination report has been prepared by this International Preliminary Examining Authority and is transmitted to the applicant according to Article 36.

2. This REPORT consists of a total of ~~3~~ 11 sheets, including this cover sheet.

   ☐ This report is also accompanied by ANNEXES, i.e., sheets of the description, claims and/or drawings which have been amended and are the basis for this report and/or sheets containing rectifications made before this Authority (see Rule 70.16 and Section 607 of the Administrative Instructions under the PCT).

   These annexes consist of a total of ___ sheets.

3. This report contains indications relating to the following items:

   - I ☒ Basis of the report
   - II ☐ Priority
   - III ☐ Non-establishment of report with regard to novelty, inventive step and industrial applicability
   - IV ☐ Lack of unity of invention
   - V ☒ Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement
   - VI ☒ Certain documents cited
   - VII ☐ Certain defects in the international application
   - VIII ☐ Certain observations on the international application

Date of submission of the demand	Date of completion of this report
28 October 2002 (28.10.2002)	31 March 2005 (31.03.2005)
Name and mailing address of the IPEA/US Mail Stop PCT, Attn: IPEA/US Commissioner for Patents P.O. Box 1450 Alexandria, Virginia 22313-1450 Facsimile No. (703) 305-3230	Authorized officer Duc T.Duong Eugenio Logan Telephone No. 571-272-2600

I. Basis of the report

1. With regard to the **elements** of the international application:*
   - [x] the international application as originally filed.
   - [x] the description:
     pages 1-25 as originally filed
     pages NONE, filed with the demand
     pages NONE, filed with the letter of ______.
   - [x] the claims:
     pages 26-29, as originally filed
     pages NONE, as amended (together with any statement) under Article 19
     pages NONE, filed with the demand
     pages NONE, filed with the letter of ______.
   - [x] the drawings:
     pages 1-13, as originally filed
     pages NONE, filed with the demand
     pages NONE, filed with the letter of ______.
   - [ ] the sequence listing part of the description:
     pages NONE, as originally filed
     pages NONE, filed with the demand
     pages NONE, filed with the letter of ______.
2. With regard to the **language**, all the elements marked above were available or furnished to this Authority in the language in which the international application was filed, unless otherwise indicated under this item.
   These elements were available or furnished to this Authority in the following language _____ which is:
   - [ ] the language of a translation furnished for the purposes of international search (under Rule23.1(b)).
   - [ ] the language of publication of the international application (under Rule 48.3(b)).
   - [ ] the language of the translation furnished for the purposes of international preliminary examination(under Rules 55.2 and/or 55.3).
3. With regard to any **nucleotide and/or amino acid sequence** disclosed in the international application, the international preliminary examination was carried out on the basis of the sequence listing:
   - [ ] contained in the international application in printed form.
   - [ ] filed together with the international application in computer readable form.
   - [ ] furnished subsequently to this Authority in written form.
   - [ ] furnished subsequently to this Authority in computer readable form.
   - [ ] The statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.
   - [ ] The statement that the information recorded in computer readable form is identical to the written sequence listing has been furnished.
4. [ ] The amendments have resulted in the cancellation of:
   - [ ] the description, pages NONE
   - [ ] the claims, Nos. NONE
   - [ ] the drawings, sheets/~~fig~~ NONE
5. [ ] This report has been established as if (some of) the amendments had not been made, since they have been considered to go beyond the disclosure as filed, as indicated in the Supplemental Box (Rule 70.2(c)).**

* *Replacement sheets which have been furnished to the receiving Office in response to an invitation under Article 14 are referred to in this report as "originally filed" and are not annexed to this report since they do not contain amendments (Rules 70.16 and 70.17).*
** *Any replacement sheet containing such amendments must be referred to under item 1 and annexed to this report.*

V. Reasoned statement under Rule 66.2(a)(ii) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1. STATEMENT

Novelty (N)	Claims	1-4 and 6-15	YES
	Claims	5	NO
Inventive Step (IS)	Claims	1-4 and 6-15	YES
	Claims	5	NO
Industrial Applicability (IA)	Claims	1-15	YES
	Claims	NONE	NO

2. CITATIONS AND EXPLANATIONS

Claim 5 lacks novelty under PCT Article 33(2) as being anticipated by Hamalainen.

Regarding to claim 5, Hamalainen discloses a communication signal transmitted via carrier wave (Fig. 3), comprising a payload portion (DATA) corresponding to at least a portion of an Internet Protocol IP packet of digital information (Fig. 4B col. 8 lines 13-20); a start frame portion (FLAG) corresponding to the payload portion, and identifying a status of the payload portion within an IP packet (Fig. 4B col. 7 lines 14-16); and error checking portion (FCS) for verifying the payload portion (Fig. 4B col. 7 lines 66-67).

Claims 1-4 and 6-15 meets the criteria set out in PCT Article 33(2)-(3), because the prior art does not teach or fairly suggest for transmitting and receiving a "frame without protocol information".

Claims 1-15 meets the criteria set out in PCT Article 33(4), and thus have industrial applicability because the subject matter claimed can be made or used in industry.

VI. Certain documents cited

1. Certain published documents (Rule 70.10)

Application No Patent No.	Publication Date *(day/month/year)*	Filing Date *(day/month/year)*	Priority date (valid claim) *(day/month/year)*
US 6,542,490 B1	01 April 2003 (01.04.2003)	29 January 1999 (29.01.1999)	None

2. Non-written disclosures (Rule 70.9)

Kind of non-written disclosure	Date of non-written disclosure *(day/month/year)*	Date of written disclosure referring to non-written disclosure *(day/month/year)*

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US 02/09830

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L29/06 H04L12/56

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L H04H H04Q

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

WPI Data, PAJ, EPO-Internal, INSPEC

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 6 065 061 A (BLAHUT DONALD EDGAR ET AL) 16 May 2000 (2000-05-16) column 1, line 14 -column 2, line 4 column 6, line 41 - line 67 column 7, line 49 - line 52	1,3-10
Y		2

X	WO 00 56018 A (NORTEL NETWORKS EUROP S A) 21 September 2000 (2000-09-21) page 1, line 6 -page 2, line 36 page 8, line 3 -page 9, line 23 claim 2	1,3-10
Y		2

	-/--	

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 February 2003	10/03/2003

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL – 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Vaskimo, K

INTERNATIONAL SEARCH REPORT

Interr Application No
PCT/US 02/09830

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	EP 1 075 118 A (NORTEL NETWORKS LTD) 7 February 2001 (2001-02-07) abstract column 1, line 5 -column 2, line 27 column 3, line 10 -column 4, line 54 column 5, line 14 - line 26 column 6, line 30 -column 7, line 11 figures 4,5	1,3-10
Y		2
Y	CARSTEN BORMANN ET AL: "RObust Header Compression (ROHC)" INTERNET ENGINEERING TASK FORCE IETF DRAFT, 'Online! 26 February 2001 (2001-02-26), pages 1-145, XP002230692 Retrieved from the Internet: <URL:http://www.globecom.net/ietf/draft/draft-ietf-rohc-rtp-09.html> 'retrieved on 2003-02-11! page 2 page 6, paragraph 1. page 21, paragraph 4.4. -page 23, paragraph 4.4.2 page 91, paragraph 5.8. page 103, paragraph 5.8.4.4. page 132, paragraph A.1.1. -page 135, paragraph A.1.3.	2
A		1,3-10
A	D. FARINACCI ET AL: "Generic Routing Encapsulation (GRE)" NETWORK WORKING GROUP REQUEST FOR COMMENTS 2784, 'Online! March 2000 (2000-03), pages 1-8, XP002231748 Retrieved from the Internet: <URL:http://www.globecom.net/ieft/rfc/rfc2784.html> 'retrieved on 2003-02-14! the whole document	1-10

INTERNATIONAL SEARCH REPORT

Information on patent family members

Intern l Application No
PCT/US 02/09830

Patent document cited in search report		Publication date	Patent family member(s)		Publication date
US 6065061	A	16-05-2000	NONE		
WO 0056018	A	21-09-2000	AU	2314100 A	04-10-2000
			EP	1163762 A1	19-12-2001
			WO	0056018 A1	21-09-2000
EP 1075118	A	07-02-2001	EP	1075118 A2	07-02-2001

INTERNATIONAL SEARCH REPORT

Inter   ıpplication No
PCT/US 02/09831

A. CLASSIFICATION OF SUBJECT MATTER
IPC 7 H04L29/06 H04L12/56

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
IPC 7 H04L H04H H04Q

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)

WPI Data, PAJ, EPO-Internal

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CARSTEN BORMANN: "RObust Header Compression (ROHC)" INTERNET ENGINEERING TASK FORCE IETF DRAFT, 'Online! 26 February 2001 (2001-02-26), pages 1-145, XP002230692 Retrieved from the Internet: <URL:http://www.globecom.net/ietf/draft/draft-ietf-rohc-rtp-09.html> 'retrieved on 2003-02-11! page 2, line 1 - line 16 page 6, paragraph 1. -page 8, paragraph 2. page 19, paragraph 4.3.1. -page 23, paragraph 4.5. ---	1-15

-/--

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

° Special categories of cited documents :

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
20 February 2003	04/03/2003

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	Vaskimo, K

INTERNATIONAL SEARCH REPORT

Inter l Application No
PCT/US 02/09831

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category °	Citation of document, with indication,where appropriate, of the relevant passages	Relevant to claim No.
A	WO 00 51308 A (ERICSSON TELEFON AB L M) 31 August 2000 (2000-08-31) page 1, line 5 -page 4, line 10 page 5, line 22 -page 6, line 4 ---	1,6, 10-13
A	US 6 032 197 A (MORAN BRIAN ET AL) 29 February 2000 (2000-02-29) column 1, line 5 -column 3, line 27 -----	1,6, 10-13

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No

PCT/US 02/09831

Patent document cited in search report		Publication date	Patent family member(s)			Publication date
WO 0051308	A	31-08-2000	AU	3578800	A	14-09-2000
			CN	1349701	T	15-05-2002
			EP	1157520	A2	28-11-2001
			JP	2002543636	T	17-12-2002
			WO	0051308	A2	31-08-2000
US 6032197	A	29-02-2000	NONE			

Network Working Group
INTERNET-DRAFT
Expires: December 2000

Carsten Bormann (ed.), TZI/Uni Bremen
xx
xx
Carsten Burmeister, Matsushita
Christopher Clanton, Nokia
Mikael Degermark, Lulea University
Hideaki Fukushima, Matsushita
Hans Hannu, Ericsson
Lars-Erik Jonsson, Ericsson
Rolf Hakenberg, Matsushita
Tmima Koren, Cisco
Khiem Le, Nokia
Zhigang Liu, Nokia
Akihiro Miyazaki, Matsushita
Krister Svanbro, Ericsson
Thomas Wiebke, Matsushita
Haihong Zheng, Nokia

June 29, 2000

RObust Header Compression (ROHC)
<draft-ietf-rohc-rtp-00.txt>

Status of this memo

This document is a product of the IETF ROHC WG. Comments should be directed to its mailing list, rohc@cdt.luth.se.

Abstract

Existing header compression schemes do not work well when used over

links with significant error rates, especially when the round-trip time of the link is long. For many bandwidth limited links where header compression is essential, such characteristics are common.

A header compression framework and a highly robust and efficient header compression scheme is introduced in this document, adaptable to the characteristics of the link over which it is used and also to the properties of the packet streams it compresses.

Table of contents

(Editor's note: The TOC has not been updated.

I have marked text I consider questionable by making it italic, and text that I think simply should be deleted by striking it through.)

1. Introduction

During the last five years, two communication technologies in particular have become commonly used by the general public: cellular telephony and the Internet. Cellular telephony has provided its users with the revolutionary possibility of always being reachable with reasonable service quality no matter where they are. However, until now the main service provided has been speech. With the Internet, the conditions have been almost the opposite. While flexibility for all kinds of usage has been its strength, its focus has been on fixed connections and large terminals, and the experienced quality of some services (such as Internet telephony) has generally been low.

Today, IP telephony is gaining momentum thanks to improved technical solutions. It seems reasonable to believe that in the years to come, IP will become a commonly used way to carry telephony. Some future cellular telephony links might also be based on IP and IP telephony. Cellular phones may have IP stacks supporting not only audio and video, but also web browsing, email, gaming, etc.

One of the scenarios we are envisioning might then be the one in Figure 1.1, where two mobile terminals are communicating with each other. Both are connected to base stations over cellular links, and the base stations are connected to each other through a wired (or possibly wireless) network. Instead of two mobile terminals, there could of course be one mobile and one wired terminal, but the case with two cellular links is technically more demanding.

```
Mobile              Base                           Base             Mobile
Terminal            Station                        Station          Terminal

       |  ~  ~  ~  \ /                           \ /  ~   ~   ~   ~   |
       |            |                             |                   |
    +--+            |                             |                 +--+
    |  |            |                             |                 |  |
    |  |            |                             |                 |  |
    +--+            |                             |                 +--+
                    |                             |
                    |=============================|

           Cellular              Wired                 Cellular
           Link                  Network               Link
```

Figure 1.1 : Scenario for IP telephony over cellular links

It is obvious that the wired network can be IP-based. With the cellular links, the situation is less clear. IP could be terminated in the fixed network, and special solutions implemented for each supported service over the cellular link. However, this would limit

the flexibility of the services supported. If technically and economically feasible, a solution with pure IP all the way from terminal to terminal would have certain advantages. However, to make IP-all-the way a viable alternative, a number of problems have to be addressed, especially regarding bandwidth efficiency.

For cellular phone systems, it is of vital importance to use the scarce radio resources in an efficient way. A sufficient number of users per cell is crucial, otherwise deployment costs will be prohibitive [CELL]. The quality of the voice service should also be as good as in today's cellular systems. It is likely that even with support for new services, lower quality of the voice service is acceptable only if costs are significantly reduced.

A problem with IP over cellular links when used for interactive voice conversations is the large header overhead. Speech data for IP telephony will most likely be carried by RTP [RTP]. A packet will then, in addition to link layer framing, have an IP [IPv4] header (20 octets), a UDP [UDP] header (8 octets), and an RTP header (12 octets) for a total of 40 octets. With IPv6 [IPv6], the IP header is 40 octets for a total of 60 octets. The size of the payload depends on the speech coding and frame sizes used and may be as low as 15-20 octets.

From these numbers, the need for reducing header sizes for efficiency reasons is obvious. However, cellular links have characteristics that make header compression as defined in [IPHC,CRTP,PPPHC] perform less than well. The most important characteristic is the lossy behavior of cellular links, where a bit error rate (BER) as high as 1e-3 must be accepted to keep the radio resources efficiently utilized [CELL]. In severe operating situations, the BER can be as high as 1e-2. The other problematic characteristic is the long round-trip time (RTT) of the cellular link, which can be as high as 100-200 milliseconds [CELL]. A viable header compression scheme for cellular links must be able to handle loss on the link between the compression and decompression point as well as loss before the compression point.

Bandwidth is the most costly resource in cellular links. Processing power is very cheap in comparison. Implementation or computational simplicity of a header compression scheme is therefore of less importance than its compression ratio and robustness.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119.

BER

Bit Error Rate. Cellular radio links have a rather high BER. In this document BER is usually given as a probability, but one also needs to consider the error distribution as bit errors are not independent. In our simulations we use a channel with a certain BER, and the error distribution is according to a realistic channel [WCDMA].

Cellular links

Wireless links between mobile terminals and base stations. The BER and the RTT are rather high in order to achieve an efficient system overall.

Compression efficiency

The performance of a header compression scheme can be described with three parameters, compression efficiency, robustness and compression reliability. The compression efficiency is determined by how much the header sizes are reduced by the compression scheme.

Compression reliability

The performance of a header compression scheme can be described with three parameters, compression efficiency, robustness and compression reliability. The compression reliability is a measure for how well the scheme ensures that the decompressed headers are not erroneous and the possibility to avoid damage propagation from the decompressor.

Context

The context is the state which the compressor uses to compress a header and which the decompressor uses to decompress a header. The context basically contains the uncompressed version of the last header sent (compressor) or received (decompressor) over the link, except for fields in the header that are included "as-is" in compressed headers or can be inferred from, e.g., the size of the link-level frame. The context can also contain additional information describing the packet stream, for example the typical inter-packet increase in sequence numbers or timestamps.

Context damage

When the context of the decompressor is not consistent with the context of the compressor, header decompression will fail. This situation can occur when the context of the decompressor has not been initialized properly or when packets have been lost or damaged between compressor and decompressor. Packets which cannot be decompressed due to inconsistent contexts are said to be lost due to context damage. Packets that are incorrectly reconstructed due to context damage are said to have suffered damage propagation.

Context repair mechanism

To avoid excessive context damage, a context repair mechanism is needed. Context repair mechanisms can be based on explicit requests for context updates, periodic updates sent by the compressor, or methods for local repair at the decompressor side.

FER

Frame Error Rate. The FER considered in this document includes the frames lost on the channel between compressor and decompressor and frames lost due to context damage. FER is here defined to be identical to packet loss rate.

(Editor: A much better definition would be to reserve FER for the error rate we get from lower layers and use PER for the error rate we generate/hand up.)

Header compression profile

A header compression profile is a specification of how to compress the headers of a certain kind of packet stream over a certain kind

of link. Compression profiles provide the details of the header compression framework introduced in this document. The profile concept makes use of profile identifiers to separate different profiles which are used when setting up the compression scheme. All variations and parameters of the header compression scheme are handled by different profile identifiers, which makes the number of profiles rather large. This can act as a deterrent when first studying the concept, but is a real strength for several reasons. One advantage of this merging of parameters into one is that new parameters can be added by the endpoints without affecting the negotiation requirements on the link in between. Another benefit of the concept is that different combinations of functionality might be implemented with different methods, meaning that the scheme can be optimized regardless of what functionality is enabled. Finally, it should be noted that even if there are a large number of profiles, only a small number of them can/will be implemented over a specific link (IPv4 and IPv6 profiles will for example probably

not coexist). Most profiles usable in a certain environment will probably also be almost identical from an implementation point of view.

Header compression CRC

A CRC (Cyclic Redundancy Checksum) computed by the compressor and included in each compressed header. Its main purpose is to provide a way for the decompressor to reliably verify the correctness of reconstructed headers. What values the CRC is computed over depends on the packet type it is included in; typically it covers most of the original header fields.

Pre-HC links

Pre-HC links are all links a packet has traversed before the header compression point. If we consider a path with cellular links as first and last hops, the Pre-HC links for the compressor at the last link are the first cellular link plus the wired links in between.

Robustness

The performance of a header compression scheme can be described with three parameters, compression efficiency, robustness and compression reliability. A robust scheme tolerates errors on the link over which header compression takes place without losing additional packets, introducing additional errors, or using more bandwidth.

RTT

Round Trip Time - The time it takes to send a packet back and forth over the link.

Simplex link

A simplex (or unidirectional) link is a point to point link without

a return channel. Over simplex links, header compression must rely on periodic refreshes since feedback from the decompressor can not be sent to the compressor. For simplex links, a header compression CRC is mandatory to guarantee correct decompression.

Spectrum efficiency

Radio resources are limited and expensive. Therefore they must be used efficiently to make the system economically feasible. In cellular systems this is achieved by maximizing the number of users served within each cell, while the quality of the provided services is kept at an acceptable level. A consequence of efficient spectrum

use is a high rate of errors (frame loss and residual bit errors), even after channel coding with error correction.

Timestamp delta

The timestamp delta is the increase in the timestamp value between two consecutive packets.

3. Background

This chapter provides a background to the subject of header compression. The fundamental ideas are described together with descriptions of existing header compression schemes, their drawbacks and requirements and motivation for new header compression solutions.

3.1. Header compression fundamentals

The main reason why header compression can be done at all is the fact that there is lots of redundancy between header fields, both within the same packet header but especially between consecutive packets belonging to the same packet stream. By sending static field information only initially and utilizing dependencies and predictability for other fields, the header size can be significantly reduced for most packets.

In general, header compression methods maintain a context, which is essentially the uncompressed version of the last header sent over the link, at both compressor and decompressor. Compression and decompression are done relative to the context. When compressed headers carry differences from the previous header, each compressed header will update the context of the decompressor. In this case, when a packet is lost between compressor and decompressor, the context of the decompressor will be brought out of sync since it is not updated correctly. A header compression method must have a way to repair the context, i.e. bring it into sync, after such events.

3.2. Existing header compression schemes

The original header compression scheme, CTCP [VJHC], was invented by Van Jacobson. CTCP compressed the 40 octet IP+TCP header to 4 octets.

The CTCP compressor detects transport-level retransmissions and sends a header that updates the context completely when they occur. This repair mechanism does not require any explicit signaling between compressor and decompressor.

CRTP [CRTP, IPHC] by Casner and Jacobson is a header compression scheme that compresses 40 octets of IPv4/UDP/RTP headers to a minimum of 2 octets when no UDP checksum is present. If the UDP checksum is present, the minimum CRTP header is 4 octets. CRTP cannot use the same repair mechanism as CTCP since UDP/RTP does not retransmit. Instead, CRTP uses explicit signaling messages from decompressor to compressor, called CONTEXT_STATE messages, to indicate that the context is out of sync. The link roundtrip time will thus limit the speed of this context repair mechanism.

On lossy links with long roundtrip times, such as most cellular links, CRTP does not perform well. Each lost packet over the link causes several subsequent packets to be lost since the context is out of sync during at least one link roundtrip time. This behavior is documented in [CRTPC]. For voice conversations such long loss events will degrade the voice quality. Moreover, bandwidth is wasted by the large headers sent by CRTP when updating the context. [CRTPC] found that CRTP performed much worse than ideally for a lossy cellular link. It is clear that CRTP alone is not a viable header compression scheme for cellular links.

To avoid losing packets due to the context being out of sync, CRTP decompressors can attempt to repair the context locally by using a mechanism known as TWICE. Each CRTP packet contains a counter which is incremented by one for each packet sent out by the CRTP compressor. If the counter increases by more than one, at least one packet was lost over the link. The decompressor then attempts to repair the context by guessing how the lost packet(s) would have updated it. The guess is then verified by decompressing the packet and checking the UDP checksum - if it succeeds, the repair is deemed successful and the packet can be forwarded or delivered. TWICE has got its name from the observation that when the compressed packet stream is regular, the correct guess is to apply the update in the current packet twice. [CRTPC] found that even with TWICE, CRTP doubled the number of lost packets. TWICE improves CRTP performance significantly. However, there are several problems with using TWICE:

1) It becomes mandatory to use the UDP checksum:

   - the minimal compressed header size increases by 100% to 4 octets.

   - most speech codecs developed for cellular links tolerate errors in the encoded data. Such codecs will not want to enable the UDP checksum, since they want damaged packets to be delivered.

   - errors in the payload will make the UDP checksum fail when the guess is correct (and might make it succeed when it is wrong).

2) Loss in an RTP stream that occurs before the compression point will make updates in CRTP headers less regular. Simple-minded versions of TWICE will then perform badly. More sophisticated versions would need more repair attempts to succeed.

3.3. Requirements on a new header compression scheme

The major problem with CRTP is that it is not sufficiently robust against packets being damaged between compressor and decompressor. A viable header compression scheme must be less fragile. This increased robustness must be obtained without increasing the compressed header

size; a larger header would make IP telephony over cellular links economically unattractive.

A major cause of the bad performance of CRTP over cellular links is the long link roundtrip time, during which many packets are lost when the context is out of sync. This problem can be attacked directly by finding ways to reduce the link roundtrip time. Future generations of cellular technologies may indeed achieve lower link roundtrip times. However, these will probably always be rather high [CELL]. The benefits in terms of lower loss and smaller bandwidth demands if the context can be repaired locally will be present even if the link roundtrip time is decreased. A reliable way to detect a successful context repair is then needed.

One might argue that a better way to solve the problem is to improve the cellular link so that packet loss is less likely to occur. It would of course be nice if the links were almost error free, but such a system would not be able to support a sufficiently large number of users per cell and would thus be economically infeasible [CELL].

One might also argue that the speech codecs should be able to deal with the kind of packet loss induced by CRTP, in particular since the speech codecs probably must be able to deal with packet loss anyway if the RTP stream crosses the Internet. While the latter is true, the kind of loss induced by CRTP is difficult to deal with. It is usually not possible to hide a loss event where well over 100 ms worth of speech is completely lost. If such loss occurs frequently at both ends of the path, the speech quality will suffer.

A detailed description of the requirements specified for ROHC may be found in [REQ].

3.4. Classification of header fields

As mentioned earlier, header compression is possible due to the fact that there is much redundancy between header field values within packets, but especially between consecutive packets. To utilize these properties for header compression, it is important to understand the behavior of the various header fields. To do this, all header fields have been classified in detail in appendix A. The fields are first classified on a high level and then some of them are studied more in detail. Finally, the appendix concludes with recommendations about how the various fields should be handled by header compression algorithms. The main conclusion that can be drawn is that most of the header fields can easily be compressed away since they are never or seldom changing. Only 5 fields with a total size of about 10 octets are rather difficult to compress and must be handled in a sophisticated way by the compression scheme. Those fields are:

- IPv4 Identification (16 bits)

- UDP Checksum (16 bits)
- RTP Marker (1 bit)
- RTP Sequence Number (16 bits)
- RTP Timestamp (32 bits)

It is rather obvious that these fields then will have a large impact on how a header compression scheme is designed. More detail about this should be found in Appendix A.

4. Header compression framework

4.1. Operating assumptions

TBW

4.2. Dynamicity

TBW

4.3. Compression and decompression states

TBW: Compression and decompression states, how they interact with each other. They must not be correlated. High level description without transitions described.

The compressor starts in the lowest compression state and gradually transitions to higher compression states. The general principle is the compressor will always operate in the highest possible compression state, under the constraint that the compressor has sufficient confidence that the decompressor has the information necessary to decompress a header compressed according to that state. In the reliable mode, that confidence comes from receipt of ACKs from the decompressor. Otherwise, that confidence comes from sending the information a certain number of times, and, if a back channel is available, from not receiving NAKs (negative acknowledgements).

The compressor may also transition back to a lower compression state when necessary.

For IP/UDP/RTP compression, the three compressor states are the Initialization/Refresh, First Order, and Second Order. A brief description of each is given in the subsections below.

4.3.1. Initialization/Refresh (IR) State

In this state, the compressor essentially sends IR headers. The information sent in a refresh may be static and non-static fields in uncompressed form (full refresh), or just non-static fields in uncompressed form (non-static refresh or dynamic refresh). The compressor enters this state at initialization, upon request from decompressor, or upon Refresh Time-out. The compressor leaves the IR state when it is confident that the decompressor has correctly received the refresh information.

4.3.2. First Order (FO) State

Subsequently to the IR state, the compressor operates in the FO state when the header stream does not conform to a uniform pattern, or when the compressor is not confident that the decompressor has acquired the parameters of the uniform pattern. In this state, the compressor essentially sends FO headers. In the case of speech with silence suppression turned on, a new talk spurt following a silence interval will result in the RTP TS incrementing by more than the regular TS increment. Consequently, the header stream does not

conform to the uniform pattern, and the compressor is in the FO state. The compressor will leave this state and transition to the SO state when the current header conforms to a string, and the compressor is confident the decompressor has acquired the parameters of the uniform pattern.

4.3.3. Second Order (SO) State

The compressor enters this state when the header to be compressed belongs to a uniform pattern, and the compressor is sufficiently confident that the decompressor has also acquired the parameters of the uniform pattern. In the SO state, the compressor sends SO headers, which mainly consist of a sequence number. While in the SO state, the decompressor does a simple extrapolation based on information it knows about the pattern of change of the header fields and the sequence number contained in the SO header in order to regenerate the uncompressed header. The compressor leaves this state to go back to FO state when the header no longer conforms to the uniform pattern.

4.4. Different modes of operation.

TBW: The difference between states and modes.

TBW: - Unidirectional mode
- Bi-directional optimistic mode

4.4.3. Bi-directional reliable mode

(Essentially Unedited Text from ACE. This is probably too long for chapter 4.)

An ACK packet contains a sequence number that uniquely identifies the compressed header packet that is being ACKed. ACKnowledgements have four main functions:

- To inform the compressor that Refresh information has been received. In that case, the compressor knows that the decompressor has acquired the information necessary to decompress FO headers. This means the compressor can reliably transition to the next higher compression state, the FO state. This kind of ACK is referred to as an IR-ACK.

- To inform the compressor that FO information has been received; in that case, the compressor knows that the decompressor has acquired the information necessary to decompress SO headers. This means the compressor can reliably transition to the next higher compression state, SO state; this kind of ACK is referred to as an FO-ACK.

- To inform the compressor that a header with a specific sequence number n has been received; in that case, the compressor knows that the decompressor can determine the sequence number without any ambiguity (caused, e.g., by counter wrap-around) up to header number n + seq_cycle, where seq_cycle is the counter cycle (determined by the number of bits, k, in the sequence number). This kind of ACK is referred to as an SO-ACK.

- When information is sent as in-band signaling, to confirm that the in-band signaling information has been received

The control of transition from IR to FO to SO states by ACKs ensures that there is no context desynchronization, and therefore no error propagation. That is, a compressed header that is not received in error can always be correctly decompressed, because synchronization is never lost.

Reception of ACKs by the compressor can also be used to increase compressor header field encoding efficiency. Compression is more efficient because the compressor just has to send the necessary information (but no more) to ensure correct decompression of the current header. In general, the minimal information that the compressor needs to send depends on what information the decompressor already knows. The information known at the decompressor is indicated to the compressor in the decompressor's ACK transmission.

An enhancement to the acknowledgement procedure can be used to reduce FO ACK traffic on the feedback channel; this traffic can be quite high if there is significant round trip delay between compressor and decompressor. In this case, several FO headers would be sent before the compressor can receive an ACK, and normally, one ACK would be sent by the decompressor for each FO header received.

The basic idea is that whenever the decompressor receives a packet and needs to send an ACK to the compressor, it just sends the ACK once (or twice if there is no default 'pattern' agreed on by the compressor and decompressor) and waits for some round trip time (as opposed to sending ACKs in response to each, e.g., FO packet on the feedback channel). After the round trip time, if the decompressor can confirm that the compressor received the ACK (evidenced by receipt of an SO packet at the decompressor), it continues normal decompression. Otherwise, it will send the ACK again and the process repeats.

The only potential negative to this approach is if the ACK sent by the decompressor is lost. In that case, progression to the next higher compression state by the compressor is delayed until the next ACK is correctly received (at least one round trip time).

The decompressor uses as reference for decompression only those headers which it is sufficiently confident of the correct decompression (secure reference). A secure reference must be chosen

from the headers received with an OK CS. Until a new secure reference is chosen, all subsequent headers are decompressed with respect to the current secure reference. A major advantage of this approach is that an undetected error which affects correct decompression of header m will not affect decompression of subsequent headers. For example, if header # 3 is an FO used as a secure reference, and header # 5 is an SO with an undetected error, the decompression of header # 6 will be based solely on header # 3 and not affected by header # 5. In other words, an undetected error will affect only the current header, just like when headers are not compressed.

4.5. Encoding methods

The analysis of header field changes in appendix A excluded changes due to loss and/or reordering before the header compression point. Such changes will have an impact on the regularity of the RTP sequence number, the RTP timestamp value and, for IPv4, the IP ID value. However, as described in A.2, both the RTP timestamp and the IP ID value (if sequentially assigned) are expected to follow the RTP sequence number for most packets. The most important task is then to communicate RTP sequence number information in an efficient way. This chapter describes the encoding methods used in a general way. How the methods are applied to fields in different compressed headers is described in the packet format chapter.

4.5.1. Least Significant Bits (LSB) encoding

A commonly used method for updating fields whose values are always subject to small changes (usually positive) is Least Significant Bits (LSB) encoding. For example, an increase of up to 16 could be handled with only 4 bits with LSB encoding (if decreases are not expected). This method is used for many different fields in the ROHC packet headers defined in this document. If a field is labeled "<fieldname> LSB", it means that the field contains only the least significant bits of the corresponding original field.

4.5.2. Least Significant Part (LSP) encoding

One restriction with LSB encoding is that whole bits are needed, meaning that only 2, 4, 8, 16, 32, ... code-points could be used. In some cases, especially when several mechanisms are integrated for efficiency reasons, it would be desirable to have a method that could make use of any number of available code-points. To signal one special event one could either use one single bit or, if the event is not to be signaled in parallel with other information, as one bit pattern for several bits. That would leave more bit patterns for other usage.

Assume that we have 4 special events to signal and 5 bits available. Taking 2 bits for these events, then there would be 3 bits (8 code-points) left for other usage. If we instead reserved 4 of the code-

points represented by all 5 bits, there would instead be 32-4=28 code-points left for other usage. The only disadvantage would be that the bits cannot be used for both purposes at the same time.

What would be desirable is to do LSB encoding of code-points instead of whole bits. Therefore the method called Least Significant Part (LSP) encoding has been introduced. LSP encoding of size (number of code-points) M for a value N is defined as:

```
  LSP:M(N) = N modulo M
```

An example showing the LSP encoding and decoding of a counter S(n) with M code-points is used below to illustrate the LSP principle. S'(n) is the decoded value corresponding to the original S(n) value. With S'(n-1), we denote the last correctly decompressed value.

```
  Input sequence: S(n)
  Encoded sequence: LSP:M(S(n)) = S(n) modulo M
  Decoded sequence: S'(n) = S'(n-1) - LSP:M(S'(n-1)) + LSP:M(S(n)) =
                            S'(n-1) - S(n-1) modulo M + S(n) modulo M
```

To handle modulo wrap-around, an additional verification is inserted. If the decoded value S'(n) is smaller than S'(n-1)-R, S'(n) is increased with M (reordering of order R is then handled with this encoding).

When applying LSP encoding, there are thus two parameters that must be set:

```
  M - The number of code-points to use (modulo value)
  R - The reordering order to handle
```

A similar mechanism as for modulo wrap-around should also be used to handle full-field wrap-around.

4.5.3. LSB or LSP encoding with extended range

If needed, it could be good to extend the range covered by the LSB or LSP encoding. For the LSB case, it is simple to send only the next more significant bits. For the LSP, what must be done is to rewrite the definition of LSP so that it defines an additional parameter.

The LSP definition from previous chapter can instead be expressed as:

```
  LSP:M(N) = N - INT:M(N)*M       [ INT:M(N) = (N - LSP:M(N)) / M) ]
```

And in that case, INT:M(N) is the integer part left after division. If additional bits can be transmitted to increase the range covered, this can be done by sending the least significant bits (LSB) of this integer part, INT:M(N). The example from previous chapter will then change into:

```
  Input sequence: S(n)
  Encoded sequence: LSP:M(S(n)) = S(n) modulo M
                    INT:M(S(n)) = (S(n)-LSP:M(S(n))) / M
```

```
Decoded sequence: S'(n) = S'(n-1) - LSB:M(S'(n-1)) * M
                                    + LSB:M(S(n)) * M
                                    - LSB(INT:M(S'(n-1))) * M
                                    + LSB(INT:M(S(n))) * M
```

4.5.4. VLE - Variable Length Encoding

[Editor: This needs to be renamed to _window-based LSB encoding_ or maybe some better term.]

An alternative approach to encoding irregular changes in header fields is to send the 'k' least significant bits of the original header field value.

Clearly, it is desirable for the compressor to minimize this number of bits. Due to the possible loss of packets on the channel between compressor and decompressor (CD-CC), the compressor does not know which packet will be used as the reference by the decompressor, and hence, does not know how many LSBs need to be sent.

Variable Length Encoding (VLE) solves this problem. The basic algorithm employs a 'sliding window', maintained by the compressor, which is advanced when the compressor has sufficient confidence that the compressor has certain information. The confidence may be obtained by various means, e.g., an ACK from the decompressor if operating in RRP. In the case of NRP a sliding window of fixed size, e.g. M (described later) may be used. In either case, the value of k determined depends on the current values in the sliding window. Details of the operation follow below.

4.5.4.1. VLE Basics

Basic concepts of VLE are:

* The decompressor uses one of the decompressed header values as a reference value, v_ref. The reference may be chosen by various means- one approach might be to select only headers whose correct reconstruction is verified by inclusion of a checksum with the compressed header ("secure" reference).

* The compressor maintains a sliding window of the values (VSW) which may be chosen as a reference by the decompressor. It also maintains the maximum value (v_max) and the minimum value (v_min) of VSW.

* When the compressor has to compress a value v, it calculates the range r = max(|v - v_max|, |v - v_min|). The value of k needed is k = ceiling(log2(2 * r + 1)), i.e., the compressor sends the ceiling(log2(2 * r +1)) LSBs of v as the encoded value.

* The compressor adds v into the VSW and updates the v_min and v_max IF the value v could potentially be used as a reference by the decompressor.

* The decompressor chooses as the decompressed value the one that

is closest to v_ref and whose k LSB equals the compressed value that has been received.

It is obvious that we need to move forward (or shrink) the sliding window to prevent k from increasing too much. To do that, the compressor only needs to know which values in VSW have been received by the decompressor. In the case of RPP, that information is carried in the ACKs. In the case of NRP, the VSW is moved without ACK, if there are a maximum number of entries, 'M', already present in VSW. M is defined in the compressor logic section and further elaborated upon in the "Implementation Hints" appendix.

The VLE concept can be applied to RTP Timestamp, RTP Sequence Number, IP-ID header fields, etc.

4.5.4.2. One-Sided Variable Length Coding (OVLE)

The VLE encoding scheme is very general and flexible, as it can accommodate arbitrary changes (positive, negative) from one value to the next. When VLE is applied to a field that is monotonic (e.g. RTP SN), there is a loss in efficiency, because k, the number of bits is defined by the condition

(2p+1) < 2 to the kth(p=|current value-reference value|).

On the other hand, if the variation is known to be monotonic, the required k is smaller, as it has to satisfy only

p < 2 to the kth.

One-Sided Variable Length Encoding (OVLE) is based on the idea to use a k that satisfies the latter condition, when the field to be compressed is monotonic (increasing or decreasing). When the field is almost always monotonic (quasi-monotonic), OVLE compression can be used when the field is behaving monotonically, and 'regular' VLE used when it is not.

The savings over VLE is 1 bit, and since that saving is achieved most of the time, it translates into a 1 bit savings in the average overhead. Alternatively, the number of bits can be kept the same, but the frequency of ACKs can be reduced by a factor of 2.

4.5.5. Timer-Based Compression of RTP Timestamp

[Editor: This needs to be aligned with 5.8.3!]

A useful observation is that the RTP timestamps when generated at the source closely follow a linear pattern as a function of the time of day clock, particularly for the case when speech is being carried in the RTP payload.

For example, if the time interval between consecutive speech samples is 20 msec, then the RTP time stamp of header n (generated at

time n*20 msec) = RTP time stamp of header 0 (generated at time 0) + TS stride * n, where TS_stride is a constant dependent on the voice codec. In what follows, n is referred to as the 'packed' RTP TS.

Consequently, the RTP TS in headers coming into the decompressor also follow a linear pattern as a function of time, but less closely, due to the delay jitter between the source and the decompressor. In normal operation (no crashes or failures), the delay jitter is bounded, to meet the requirements of conversational real-time traffic. Thus, it is possible for the decompressor to obtain an approximation of the packed RTP TS of the current header (the one to

be decompressed), by adding the time elapsed since a reference header to the packed RTP TS of that reference header. The decompressor then refines this approximation with the additional information received in the compressed header. The compressed header carries the k least significant bits of the packed RTP TS. The required value of k to ensure correct decompression is a function of the jitter between the source and decompressor. The compressor can estimate the jitter and determine k, or alternatively it can have a fixed k,
and filter out the packets with excessive jitter. Once the decompressor has the packed RTP TS, it can convert to the original RTP TS.

The advantages to this approach are many:

* The size of the compressed RTP TS is constant and small. In particular, it does NOT depend on the length of the silence interval. This is in contrast to other RTP TS compression techniques, which require a variable number of bits dependent on the duration of the preceding silence interval. It is very important to be able to efficiently compress the RTP TS, as it is one of the essential changing fields (see Appendix A).

* No synchronization is required between the timer process and the decompressor process.

* Robustness to errors: the partial RTP TS information in the compressed header is self contained and only needs to be combined with the decompressor timer to yield the full RTP TS value. Loss or corruption of a header will not invalidate subsequent compressed headers.

As an example, consider the scenario in which a long silence interval has just ended, and the header compressor scheme is preparing to send an FO header to decompressor to adjust for the unexpected change in RTP timestamp. The compressor knows that the packet which has just arrived is the first packet of a new talkspurt as opposed to following a lost packet because the RTP SN increments by only one. Note that we need not assume any special behavior of the input to the compressor (i.e. the scheme tolerates reordering, or more generally, non-increasing RTP timestamp behavior observed prior to the compressor).

At the end of the silence interval, the compressor sends in the FO

compressed header the k least significant bits of

p_TS_current = (current RTP time stamp - TS0)/TS_stride.

p_TS_current is the "packed" representation of the current time; it has granularity of TS_stride, which is the RTP timestamp increment

observed during e.g. a VoIP session (e.g. 160 for a 20 mS voice codec).

TS0 is an arbitrary timestamp offset.

The compressor runs the following algorithm to determine k.

STEP 1: calculate Network_Jitter (Current_header, j) as

| (T_current - T_j) - (p_TS_current - p_TS_j) |

for all packets in a sliding window, TSW. TSW contains several pairs (T_j, p_TS_j) of values corresponding to the packets sent that may be used as a reference, including the last packet which was ACKed. In the case of RPP, TWS is moved when an ACK with some indication is received from the decompressor. In the case of NRP mode, the TSW is moved without ACK if there are a maximum number of entries, 'M', present in TSW. I.e., the sliding window is managed just like for the case of VLE.

T_current is the current wall clock time at the compressor, and T_j is the wall clock time at which the packet j in the sliding window was received by the compressor. Both T_current and T_j are in units of time interval (e.g. 20 ms) equivalent to TS_stride.

p_TS_current and p_TS_j are the packed RTP timestamp times of the packets, determined from the actual RTP header.

STEP 2: compute Max_Network_Jitter, where

Max_Network_Jitter = Max(Network_Jitter(current, j)), for all headers j in TSW

Note that Max_Network_Jitter is positive.

STEP 3: k is then calculated as

k = ceiling(log2(2 * J + 1)), where

J = Max_Network_Jitter + Max_CD_CC_Jitter + 2.

Max_CD_CC_Jitter is the maximum possible CD-CC jitter expected on the CD-CC. Such a maximum depends only on the characteristics of the CD- CC, and is expected to be reasonably small in order to have good quality for real-time services.

The factor + 2 is to account for the quantization error caused by the timers at the compressor and decompressor, which can be +/- 1.

4.6. Requirements on lower layers

This chapter lists general ROHC requirements on an underlying link layer. See also the ROHC lower layer requirements document [LLG].

Framing

Framing, which makes it possible to separate different packets, is the most important link layer functionality.

Length

Most link layers can indicate the length of the packet, and this information has therefore been removed from the packet headers. This means that it now MUST be given by the link layer.

Error protection

A suitable link layer CRC covering at least the header part of the packet is assumed. The CRC SHOULD ensure that packets with errors in the header part are never delivered.

Negotiation

In addition to the packet handling mechanisms above, the link layer MUST also provide a way to carry out the negotiation of header compression parameters.

5. The protocol

5.1. Data structures

5.1.1. Per-channel parameters

5.1.2. Per-CID parameters, profiles

5.1.3. Contexts

5.2. Packet types

This chapter defines the various packet types that are used by the ROHC scheme. It also lists some parameters that are needed in the various packet headers to carry out transition between the three modes of operation.

5.2.1. Packets from compressor to decompressor

The ROHC scheme defines three different packet types for the information sent from compressor to decompressor. The header formats for these packet types may vary but their meaning will always be the same. The three packet types defined are:

FH (Full Header) : In these packets, all information needed to establish the decompressor context is sent.

FO (First Order) : Only a limited amount of context information is sent in a FO packet and no STATIC information. However, successful decompression of subsequent packets requires that the information sent in an FO packet is correctly transferred.

SO (Second Order) : The SO packets are small and (almost) independent packets. Subsequent packets are not depending on the SO packet for successful decompression.

TBW: how these packet types are used, their relation to the three compression states with the same names etc...

5.2.2. Feedback packets from decompressor to compressor

In addition to the packet types used from compressor to decompressor,

feedback packets are also defined to use from decompressor to compressor. The feedback packet formats may vary and there may also be special feedback packet types defined. However, these three feedback packet types must always be supported to control state and mode transition:

```
ACK         : Acknowledge a successful decompression of a packet,
              which means that the context is up to date.

NACK        : Indicates that decompression has failed.

STATIC-NACK : Indicates that decompression has failed due to an
              invalid (or never established) STATIC context.
```

TBW: How these packet types are used etc...

5.2.3. Parameters needed for mode transition

TBW:

All feedback packets of the types defined above must carry the sequence number of the packet that it corresponds to and a parameter telling the desired compression mode to work in (U=Unidirectional, O=Optimistic, R=Reliable).

5.3. Operation in unidirectional mode

TBW: General description of the unidirectional mode

5.3.1. Compression states and logic

Below is the state machine for the compressor in unidirectional mode. Details of each state, the transitions between states and compression logic is given subsequent to the figure.

```
              Optimistic approach         Optimistic approach
             +------>------>------+      +------>------>------+
             |                    |      |                    |
             |                    v      |                    v
+----------+                     +----------+               +----------+
| FH State |                     | FO State |               | SO State |
+----------+                     +----------+               +----------+
   ^       ^                      |      ^                    |       |
   |       |       Timeout        |      |  Timeout / Update  |       |
   |       +------<------<------+        +------<------<------+       |
   |                                                                  |
```



```
   |                            Timeout                               |
   +------<------<------<------<------<------<------<------<------<------+
```

TBW: Descriptions of timers, optimistic approach, transitions and the packets used. Text from ROCCO draft may be reused.

5.3.2. Decompression states and logic

```
                                     FH
               +-->------->------->------->------->------->-+
               |                                             |
    FO,SO      |              SO           FH,FO      |    FH,FO,SO
   +-->--+     |           +-->--+        +-->------->-+   +-->--+
   |     |     |           |     |        |            |   |     |
   |     v     |           |     v        |            v   |     v
 +---------------+       +-----------------+         +-----------------+
 |  No Context   |       | Static Context  |         |  Full Context   |
 +---------------+       +-----------------+         +-----------------+
      ^                         |       ^                         |
      | Invalid decompression   |       | Invalid decompression   |
      +-------<-------<-------<-+       +-----<-------<-------<---+
```

TBW: CRC failure, repeated reconstruction's, decompressor sliding window, timer-based decompression, transition logic

No feedback messages are sent to the compressor when working in unidirectional mode.

[illegible] Operation in bi-directional optimistic mode

TBW: Description of this mode

[illegible] Compression states and logic

[illegible] is the state machine for the compressor in bi-directional optimistic mode. Details of each state, the transitions between states and compression logic is given subsequent to the figure.

```
                     Optimistic approach / ACK
   +------>------->------->------->------->------->------->------->------+
   |                                                                     |
   |        Optimistic appr. / ACK        Optimistic appr. / ACK         |
   |       +------>------->------+       +------->------->-------+       |
   |       |                     |       |                       |       |
   |       |                     v       |                       v       v
```



```
 +----------+              +----------+              +----------+
 | FH State |              | FO State |              | SO State |
 +----------+              +----------+              +----------+
   ^     ^                      |       ^                 |      |
   |     |     STATIC-NACK      |       |  NACK / Update  |      |
   |     +------<-------<-------+       +------<------<---+      |
   |                                                             |
   |                          STATIC-NACK                        |
   +------<-------<-------<-------<-------<-------<-------<------+
```

TBW: Descriptions of optimistic approach, transitions and the packets used.

5.4.2. Decompression states and logic

The decompression states and the state transition logic are the same as for the unidirectional case (see section 5.3.2). What differs is the feedback logic, which states what feedback messages to send due to different events when operating in the various states.

In NC state:
- When a FH packet is correctly decompressed, send an ACK with the mode parameter set to 0
- When an FO or SO packet is received or decompression of a FH packet has failed, send a STATIC-NACK with the mode parameter set to 0

In SC state:
- When a FH packet is correctly decompressed, send an ACK with the mode parameter set to 0
- When an FO packet is correctly decompressed, optionally send an ACK with the mode parameter set to 0
- When a SO packet is received, send a NACK with the mode parameter set to 0
- When decompression of an FO or FH packet has failed, send a STATIC-NACK with the mode parameter set to 0

In FC state:
- When a FH packet is correctly decompressed, send an ACK with the mode parameter set to 0
- When an FO packet is correctly decompressed, optionally send an ACK with the mode parameter set to 0
- When an SO packet is correctly decompressed, no feedback should be sent
- When decompression of an SO, FO or FH packet has failed, send a NACK with the mode parameter set to 0

5.5. Operation in bi-directional reliable mode

5.5.1. Compression states and logic

Below is the state machine for the compressor in bi-directional reliable mode. Details of each state, the transitions between states and compression logic is given subsequent to the figure.

```
                                ACK
   +------->------->------->------->------->------->------->-------+
   |                                                               |
   |            ACK                            ACK                 |   ACK
   |       +------->------->-------+       +------->------->-------+  +->-+
   |       |                       |       |                       |  |   |
   |       |                       v       |                       v  |   v
 +------------+               +-------------+                   +----------+
```

```
  | FH State |                  | FO State |                | SO State |
  +----------+                  +----------+                +----------+
     ^      ^                        |     ^                     |      |
     |      |     STATIC-NACK        |     |   NACK / Update     |      |
     |      +-------<-------<--------+     +-------<-------<-----+      |
     |                                                                  |
     |                                                                  |
     |                           STATIC-NACK                            |
     +------<------<------<------<------<------<------<------<------<---+
```

TBW: Descriptions of transitions and the packets used.

5.5.2. Decompression states and logic

The decompression states and the state transition logic are the same as for the unidirectional case (see section 5.3.2). What differs is the feedback logic, which states what feedback messages to send due to different events when operating in the various states.

In NC state: - When a FH packet is correctly decompressed, send an ACK with the mode parameter set to R
- When an FO or SO packet is received or decompression of a FH packet has failed, send a STATIC-NACK with the mode parameter set to R

In SC state: - When an FO or FH packet is correctly decompressed, send an ACK with the mode parameter set to R
- When an SO packet is received, send a NACK with the mode parameter set to R
- When decompression of an FO or FH packet has failed, send a STATIC-NACK with the mode parameter set to R

In FC state: - When an FO or FH packet is correctly decompressed, send an ACK with the mode parameter set to R
- When an updating SO packet is correctly decompressed, periodically send an ACK with the mode parameter set to R
- When decompression of an SO, FO or FH packet has failed, send a NACK with the mode parameter set to R

5.6. Mode transitions

The decision to move from one compression mode to another is taken by the decompressor and the possible mode transitions are shown in the figure below. How the transitions are performed is described in the subsequent chapters.

```
                 +----------------------+
                 | Unidirectional mode  |
                 +----------------------+
                    ^                |
                    |                |
```

```
          |                 v
     +----------------------+
     |   Optimistic mode    |
     +----------------------+
          ^                 |
          |                 |
          |                 v
     +----------------------+
     |    Reliable mode     |
     +----------------------+
```

5.6.1. From Unidirectional to Optimistic mode

As long as there is a feedback channel available, the decompressor can at any moment decide to initiate transition from unidirectional to bi-directional Optimistic mode. All feedback packets can be used with the mode parameter set to O=Optimistic and the decompressor can then directly start working in Optimistic mode.

ACK (O) : Is sent to transit after a successful decompression. The compressor can, when receiving this packet, move directly to SO state if no update is needed compared to the acknowledged packet.

NACK (O) : Is sent to transit after a decompression failure if any preceding packet has been correctly decompressed. The compressor must, when receiving

this packet, go to FO state to update the decompressor context.

STATIC-NACK (O) : is sent to transit after a decompression failure when decompression has never succeeded. The compressor must, when receiving this packet, start from FH state to establish the static part of the context.

5.6.2. From Optimistic to Reliable mode

Transition from Optimistic to Reliable mode is only allowed after at least one packet has been correctly decompressed, which means that the static part of the context is established. Either the ACK(R) or the NACK(R) feedback packet is used to initiate the transition and the compressor MUST always run in FO state during transition. The transition procedure is described below:

```
  Compressor                          Decompressor
  ----------------------------------------------------
                                        |
          |                             |
          |      ACK(R)/NACK(R) +-<-<-<-|  Transition
          |      +-<-<-<-<-<-<-<-+      |  initiated
          |-<-<-<-+                     |
```

```
Go to FO state   |                                    |
                 |->->->-+    FO(SN0,R)               |
                 |        +->->->->->->->->->-+       |
                 |->->.                   +->->->-|   Decompressor
                 |->-..                           |   transits to
                 |         ACK(SN0,R)   +-<-<-<-|    Reliable mode
                 |        +-<-<-<-<-<-<-<-+       |
Transition       |-<-<-<-+                        |
completed        |                                |
                 |->->->-+ SO (Reliable mode)     |
                 |        +->->->->->->->-+       |
                 |                        +->->->-|
                 |
```

As long as the decompressor has not received an FO packet with the mode transition parameter set to R, it must stay in Optimistic mode. The compressor must stay in FO state until it has received an ACK for an FO packet sent with the mode transition parameter set to R (indicated by the sequence number).

Since transition from Unidirectional to Optimistic mode do not require any handshakes, it is possible to transit directly from Unidirectional to Reliable mode, but only if at least one packet has been successfully decompressed indicating a correct static context at the decompressor.

5.6.3. From Reliable to Optimistic mode

Either the ACK(O) or the NACK(O) feedback packet is used to initiate the transition from Reliable to Optimistic mode and the compressor MUST always run in FO state during transition. The transition procedure is described below:

```
                 Compressor                     Decompressor
             ----------------------------------------------------
                 |                                    |
                 |      ACK(O)/NACK(O) +-<-<-<-<-|  Transition
                 |       +-<-<-<-<-<-<-<-<-+          |  initiated
                 |-<-<-<-+                            |
Go to FO state   |                                    |
                 |->->->-+    FO(SN0,O)               |
                 |       +->->->->->->->-+            |
                 |->-..                     +->->->-|  Decompressor
                 |->-..                               |  transits to
                 |         ACK(SN0,O)   +-<-<-<-|   Optimistic mode
                 |       +-<-<-<-<-<-<-<-<-+          |
Transition       |-<-<-<-+                            |
completed        |                                    |
                 |->->->-+ SO (Optimistic mode)       |
                 |       +->->->->->->->-+            |
                 |                         +->->->-|
                 |                                    |
```

As long as the decompressor has not received an FO packet with the mode transition parameter set to 0, it must stay in Reliable mode. The compressor must stay in FO state until it has received an ACK for a FO packet sent with the mode transition parameter set to 0 (indicated by the sequence number).

5.6.4. From optimistic to unidirectional mode

TBW: (idea text provided at the moment)

Initiate at compressor side by sending FO/FH packets with NO_FEEDBACK flag set. When ack of that FO/ FH is received with mode-flag set to U:

* for optimistic mode, go to unidirectional.
* for reliable mode, go to FO state and start transition procedure to optimistic mode, but with the FO mode parameter set to U. When procedure has completed, go to unidirectional mode.

In both cases, SO packets in the forward direction indicates to the decompressor that the transition has completed.

Transition could also be initiated by the decompressor by sending U-marked feedback. Decompressor could stop sending feedback when it receives periodic refreshes from compressor.

5.7. Packet formats

Table 5.1 describes which packet formats are used.

NOTE: These packet formats do not include the parameters needed for mode transition, those must be added for the scheme to work.

The first five columns profile state parameters that affect the choice of packet types:

IPv This is the IP version for which the profile is designed. Possible values for this column are 6 and 4.

CID This column gives the number of concurrent packet streams that are supported by the header compression profile through context identifiers (CIDs).

Chk This column indicates whether the profile supports packet streams with the UDP checksum (E)nabled or D(isabled).

ID For profiles supporting IPv4, this column indicates which behavior of the IPv4 Identification field the profile is optimized for. Possible values in this column are:

(S)EQUENTIAL These profiles can handle all kind of Identification assignment methods but will be less efficient than RANDOM profiles if the assignment truly is random. If the value is sequentially assigned, no extra overhead is added

for Identification.

(R)ANDOM These profiles are recommended if it is known that random assignment is used. The Identification field will be included "as-is" which means that the header size will increase by two octets.

TbT Timer-based Timestamp decompression. Requires a timer at the decompressor side to estimate timestamp jumps. Compressor never sends more than a few bits of timestamp LSB with these profiles. Can be (E)nabled or (D)isabled (see chapter 5.5.3).

S S gives the minimal header size for the profile.

The next five columns indicate how each profile is implemented. This includes header formats for STATIC (STA, see chapter 5.7.1), DYNAMIC (DYN, see chapter 5.7.2) and COMPRESSED (COM, see chapter 5.7.3) packets, and also what EXTENSION (EXT, see chapter 5.7.4) formats are used with the COMPRESSED packets. The CRC column tells the coverage of the header compression CRC: uncompressed (H)eader or the same coverage as for the UDP (C)hecksum (see chapter 5.8.).

I	C	C	I	T	S	S	D	C	E	C
P	I	h	D	B		T	Y	O	X	R
v	D	k		T		A	N	M	T	C
6	1	E	-	E	2	1	1	1	A	C
6	1	E	-	D	2	1	1	1	A	C
6	256	E	-	E	3	2	2	2	A	C
6	256	E	-	D	3	2	2	2	A	C
4	1	D	S	E	1	3	3	5/17, 9	D	R
4	1	D	S	D	1	3	3	5/17,13	B	R
4	1	D	R	E	3	3	3	7/19,11	C	H
4	1	D	R	D	3	3	3	7/19,15	A	H
4	1	E	S	E	2	3	5	1	D	C
4	1	E	S	D	2	3	5	1	B	C
4	1	E	R	E	4	3	5	3	C	C
4	1	E	R	D	4	3	5	3	A	C
4	256	D	S	E	2	4	4	6/18,10	D	H
4	256	D	S	D	2	4	4	6/18,14	B	H
4	256	D	R	E	4	4	4	8/20,12	C	H
4	256	D	R	D	4	4	4	8/20,16	A	H
4	256	E	S	E	3	4	6	2	D	C
4	256	E	S	D	3	4	6	2	B	C
4	256	E	R	E	5	4	6	4	C	C
4	256	E	R	D	5	4	6	4	A	C

```
+---+-----+---+---+---+   +---+   +-----+-----+---------+---+---+
```

Table 5.1 : Packet format table

The packet types defined in chapter 5.2 are implemented with 4 different packet formats: STATIC, DYNAMIC, COMPRESSED and FEEDBACK.

To identify the packet format used, 4 bit patterns for the initial 5 bits of the first octet (not including a potential CID field) in all packets are reserved. These patterns are:

```
STATIC       11111
DYNAMIC      1110*      (both 11100 and 11101 are reserved for this)
FEEDBACK     11110
```

The other 28 (32-4) bit patterns indicate a COMPRESSED packet format and the usage of these patterns are explained further on.

This section defines the header formats of the four ordinary packet formats STATIC, DYNAMIC, COMPRESSED and FEEDBACK together with descriptions of when and how to use them. A subsections is also dedicated to the EXTENSION formats of COMPRESSED headers.

5.7.1. Static information packets, initialization

The STATIC packet type is a packet containing no payload but only the header fields that are expected to be constant throughout the lifetime of the packet stream (classified as STATIC in appendix A). A packet of this kind MUST be sent once as the first packet from compressor to decompressor and also when requested by the decompressor (see chapter 5.4.5). If the D-bit is set, a DYNAMIC packet (without CID) is attached to the STATIC packet to create a complete context initialization packet. The STATIC packet formats are shown below for IPv6 and IPv4, respectively. Note that some fields are only present in some of the STATIC packet types.

IPv6 (45-46 octets): STATIC1, STATIC2:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
:   Context Identifier  (CID)   :   only present in STATIC2
+---+---+---+---+---+---+---+---+
| 1   1   1   1   1 | D | - | - |
+---+---+---+---+---+---+---+---+
|                               |
+           Flow Label          +
|                               |
+               +---+---+---+---+
                | - | - | P | S |
+---+---+---+---+---+---+---+---+
|                               |
/         Source Address        /   16 octets
|                               |
+---+---+---+---+---+---+---+---+
|                               |
/      Destination Address      /   16 octets
|                               |
+---+---+---+---+---+---+---+---+
|                               |
+          Source Port          +
|                               |
+---+---+---+---+---+---+---+---+
|                               |
+        Destination Port       +
|                               |
+---+---+---+---+---+---+---+---+
|                               |
/              SSRC             /   4 octets
|                               |
+---+---+---+---+---+---+---+---+
|    Header Compression CRC     |   see chapter 5.9.1.
+---+---+---+---+---+---+---+---+
```

IPv4 (18-19 octets): STATIC3, STATIC4:

```
    0   1   2   3   4   5   6   7
  +...+...+...+...+...+...+...+...+
  :   Context Identifier  (CID)   :  only present in STATIC4
  +---+---+---+---+---+---+---+---+
  | 1   1   1   1   1 | F | P | E |
  +---+---+---+---+---+---+---+---+
  |                               |
  /        Source Address         /  4 octets
  |                               |
  +---+---+---+---+---+---+---+---+
  |                               |
  /      Destination Address      /  4 octets
  |                               |
  +---+---+---+---+---+---+---+---+
  |                               |
  +          Source Port          +
  |                               |
  +---+---+---+---+---+---+---+---+
  |                               |
  +       Destination Port        +
  |                               |
  +---+---+---+---+---+---+---+---+
  |                               |
  /             SSRC              /  4 octets
  |                               |
  +---+---+---+---+---+---+---+---+
  |    Header Compression CRC     |  see chapter 5.9.1
  +---+---+---+---+---+---+---+---+
```

All fields except for the initial five bits, the padding (-) and the Header Compression CRC are the ordinary IP, UDP and RTP fields (F=IPv4 May Fragment, P=RTP Padding, E=RTP Extension).

The number of STATIC packets sent on each occasion should be limited. If the decompressor receives DYNAMIC or COMPRESSED headers without having received a STATIC packet, the decompressor MUST send a STATIC_FAILURE_FEEDBACK packet.

5.7.2. Dynamic information packets

The DYNAMIC packet type has a header containing all changing header fields in their original, uncompressed form, and carries a payload just like ordinary COMPRESSED packets. This packet type is used after the initial STATIC packet to set up the decompressor context for the first time, and also whenever the header field information cannot be

encoded in EXTENDED_COMPRESSED packets. DYNAMIC packets could be used due to significant field changes or upon INVALID_CONTEXT_FEEBACK.

All fields except for the initial four bits, the Timestamp Delta, and the Header Compression CRC are ordinary IP, UDP and RTP fields. The Timestamp Delta is the current delta between RTP timestamps in consecutive RTP packets. Initially this value SHOULD be set to 160.

The packet formats are shown below for IPv6 and IPv4, respectively. Note that some fields are only present in some of the DYNAMIC packet types.

IPv6 (13-16 octets + CSRC List of 0-60 octets): DYNAMIC1, DYNAMIC2:

```
  0   1   2   3   4   5   6   7
+...+...+...+...+...+...+...+...+
:   Context Identifier  (CID)   :  only in DYNAMIC2
+---+---+---+---+---+---+---+---+
| 1   :   1   0 |  CSRC Counter |
+---+---+---+---+---+---+---+---+
|          Traffic Class        |
+---+---+---+---+---+---+---+---+
|            Hop Limit          |
+---+---+---+---+---+---+---+---+
|                               |
+          UDP Checksum         +
|                               |
+---+---+---+---+---+---+---+---+
| M |        Payload Type       |
+---+---+---+---+---+---+---+---+
|                               |
+        Sequence Number        +
|                               |
+---+---+---+---+---+---+---+---+
|                               |
/           Timestamp           /  4 octets
|                               |
+---+---+---+---+---+---+---+---+
:                               :
:           CSRC List           :  0-15 x 4 octets
:                               :
+---+---+---+---+---+---+---+---+
|                               |
+        Timestamp Delta        +
|                               |
+---+---+---+---+---+---+---+---+
|    Header Compression CRC     |  see chapter 5.9.2.
+---+---+---+---+---+---+---+---+
/            Payload            /
+---+---+---+---+---+---+---+---+
```


IPv4 (15-18 octets + CSRC List of 0-60 octets): DYNAMIC3, DYNAMIC4, DYNAMIC5, DYNAMIC6:

```
  0   1   2   3   4   5   6   7
+...+...+...+...+...+...+...+...+
:   Context Identifier  (CID)   :  only in DYNAMIC4 and DYNAMIC6
+---+---+---+---+---+---+---+---+
```

```
| 1   1   1   0 | CSRC Counter |
+---+---+---+---+---+---+---+---+
|       Type Of Service         |
+---+---+---+---+---+---+---+---+
|                               |
+       Identification          +
|                               |
+---+---+---+---+---+---+---+---+
|         Time To Live          |
+---+---+---+---+---+---+---+---+
:                               :
+         UDP Checksum          +  only in DYNAMIC5 and DYNAMIC6
:                               :
+---+---+---+---+---+---+---+---+
| M |        Payload Type       |
+---+---+---+---+---+---+---+---+
|                               |
+       Sequence Number         +
|                               |
+---+---+---+---+---+---+---+---+
|                               |
/          Timestamp            /  4 octets
|                               |
+---+---+---+---+---+---+---+---+
:                               :
:           CSRC List           :  0-15 x 4 octets
:                               :
+---+---+---+---+---+---+---+---+
|                               |
+           TS Delta            +
|                               |
+---+---+---+---+---+---+---+---+
|   Header Compression CRC      |  see chapter 5.8.2.
+---+---+---+---+---+---+---+---+
/           Payload             /
+---+---+---+---+---+---+---+---+
```

Each time a DYNAMIC packet is sent, several subsequent packets SHOULD also be DYNAMIC packets to ensure a successful update even when packets are lost. Context updates both with DYNAMIC and COMPRESSED packets could also be acknowledged with CONTEXT_UPDATED_FEEDBACK.

5.7.3. Compressed packets

The COMPRESSED packet type is the most commonly used packet and is designed to handle ordinary changes as efficiently as possible.

When changes are regular, all information is carried in the base header. When desired, it is possible to send additional information in extensions to the COMPRESSED base-header.

The COMPRESSED base-header formats are shown below. Note that some fields are only present in some of the COMPRESSED packet types.

```
Defines packet types: COMPRESSED1..COMPRESSED4:

      0   1   2   3   4   5   6   7
    +...+...+...+...+...+...+...+...+
    :   Context Identifier  (CID)   : only in COMPRESSED type 2 and 4
    +---+---+---+---+---+---+---+---+
    |   Sequence LSB#   |           |     # see chapter 5.8.2
    +---+---+---+---+---+       +---+
    |   Header Compression CRC* | X |     * see chapter 5.9.3
    +---+---+---+---+---+---+---+---+
    :                               :
    +         Identification        + only in COMPRESSED type 3 and 4
    :                               :
    +...+...+...+...+...+...+...+...+
    :                               :
    /           Extension           / only present if X=1
    :                               :
    +---+---+---+---+---+---+---+---+
    /             Payload           /
    +---+---+---+---+---+---+---+---+


Defines packet types: COMPRESSED5..COMPRESSED8:

      0   1   2   3   4   5   6   7
    +...+...+...+...+...+...+...+...+
    :   Context Identifier  (CID)   : only in COMPRESSED 6 and 8
    +---+---+---+---+---+---+---+---+
    | 0 | Sequence LSB#     CRC*    |     # see chapter 5.8.1
    +---+---+---+---+---+---+---+---+     * see chapter 5.9.3
    :                               :
    +         Identification        + only in COMPRESSED 7 and 8
    :                               :
    +---+---+---+---+---+---+---+---+
    /             Payload           /
    +---+---+---+---+---+---+---+---+
```



```
Defines packet types: COMPRESSED9..COMPRESSED12:

      0   1   2   3   4   5   6   7
    +...+...+...+...+...+...+...+...+
    :   Context Identifier  (CID)   : only in COMPRESSED 10 and 12
    +---+---+---+---+---+---+---+---+
    | 1   0 |         CRC*          |     * see chapter 5.9.3
    +---+---+---+---+---+---+---+---+
    | Sequence LSB# |  STS LSB# | X |     # see chapter 5.8.1
    +---+---+---+---+---+---+---+---+
    :                               :
    +         Identification        + only in COMPRESSED 11 and 12
    :                               :
    +...+...+...+...+...+...+...+...+
    :                               :
```

```
/             Extension           / only present if X=1
:                                 :
+---+---+---+---+---+---+---+---+
/              Payload            /
+---+---+---+---+---+---+---+---+
```

Defines packet types: COMPRESSED13..COMPRESSED16:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
:   Context Identifier  (CID)   : only in COMPRESSED 14 and 16
+---+---+---+---+---+---+---+---+
| 1   0 | M |       STS LSB#    |     # see chapter 5.8.1
+---+---+---+---+---+---+---+---+
| Sequence LSB# |    CRC*   | X |     * see chapter 5.9.3
+---+---+---+---+---+---+---+---+
:                               :
+         Identification        + only in COMPRESSED 15 and 16
:                               :
+---+---+---+---+---+---+---+---+
:                               :
/           Extension           / only present if X=1
:                               :
+---+---+---+---+---+---+---+---+
/            Payload            /
+---+---+---+---+---+---+---+---+
```


Defines packet types: COMPRESSED17..COMPRESSED20:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
:   Context Identifier  (CID)   : only in COMPRESSED 18 and 20
+---+---+---+---+---+---+---+---+
| 0 | C |     Sequence LSB#     |     # see chapter 5.8.1
+---+---+---+---+---+---+---+---+
:             CRC*              : only present if C=1
+---+---+---+---+---+---+---+---+     * see chapter 5.9.3
:                               :
+         Identification        + only in COMPRESSED 19 and 20
:                               :
+---+---+---+---+---+---+---+---+
/            Payload            /
+---+---+---+---+---+---+---+---+
```

The coverage of the Header Compression CRC is described in chapter 5.9.3. In that chapter, the CRC polynomials to use are also defined.

The interpretations of the Sequence and STS (Scaled TimeStamp) fields

for different packet formats are given in section 5.8.

5.7.4. Extensions to compressed headers

Less regular changes in the header fields or updates of decompressor contexts require an extension in addition to the base header. When there is an extension present in the COMPRESSED packet, this is indicated by the extension bit (X) being set. Extensions are of variable size depending on the information needed to be transmitted. However, the first three extension bits are used as an extension Type field for all extension formats. The extension can carry an M-bit, a t-bit, a SEQ EXT LSB field (called SEQ*), a (S)TS (EXT) LSB field (called TS*), an ID LSB field and a bit mask for additional fields. The M-bit is the RTP marker bit and the (S)TS (EXT) LSB is timestamp information sent with the least significant bits (the most significant bits are then expected to be unchanged compared to context). The timestamp information could either be the LSB of the (S)caled (T)ime(S)tamp value (if indicated with the t-bit unset) or the LSB of the absolute timestamp value. For profiles with a timestamp field in the compressed base header, the timestamp information is sent as an extended range to that field. The SEQ EXT LSB is extended range for the RTP sequence number. How extended range works is described in chapter 5.5.1 and 5.5.2. The t-bit is sent when timestamp is not scaled, otherwise it is always scaled with the timestamp delta. The ID LSB is the LSB of the IP Identification value. Various bit mask patterns are possible and can consist of S,H,C,D,T and I. The interpretations of these bits are given at the end of this chapter.

The guiding principle for choosing the extension type is to find the smallest header type that can communicate the information needed.

For the profiles defined in this document, four different extension sets are used, called A, B, C and D. Set A and C are for IPv6 and do not handle the IPv4 identification field, which set B and D do. Set A and B handle timestamp information which set C and D do not. All possible extensions are shown below with indications of which sets and types the extensions correspond to. For instance, B3 means that in extension set B, the extension is used with type value 3 (indicated in the type field).

The defined extension types are shown below:

```
                    0             7
                - - +-+-+-+-+-+-+-+-+
 A0, B0,            |0 0 0    SEQ*  |
 C0, D0         - - +-+-+-+-+-+-+-+-+

                    0             7
                - - +-+-+-+-+-+-+-+-+
 A1, B1             |0 0 1|M|  TS*  |
                - - +-+-+-+-+-+-+-+-+
```



```
                                            1
                    0             7 8       5
                - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 A2                 |0 1 0|M|          TS*          |
                - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                            1 1             2
                    0             7 8       5 6             3
                - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 A3                 |0 1 1|M|t|               TS*                   |
                - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                    0             7
                - - +-+-+-+-+-+-+-+-+ - -
 A4                 |1 0 0|M|R|D|T|t|
                - - +-+-+-+-+-+-+-+-+ - -

                                            1
                    0             7 8       5
                - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
 A5                 |1 0 1|M|C|R|S|D|      TS*      |
                - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -

                                            1 1             2
                    0             7 8       5 6             3
                - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
 A6                 |1 1 0|M|C|R|S|D|t|           TS*               |
                - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
```

```
                                                1
                 0               7 8            5
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
    A?           |1 1 1|M|C|H|S|D|T|t|    SEQ*   |
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -

                                                1
                 0               7 8            5
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    B2           |0 1 0|M|      TS*      |  SEQ' |
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                                1
                 0               7 8            5
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    B3           |0 1 1|M|  TS*  |     ID LSB    |
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Bormann (ed.)   [Page 46]

INTERNET-DRAFT   Robust Header Compression   June 29, 2000

```
                 0               7
             - - +-+-+-+-+-+-+-+-+
    B4, D4       |1 0 0|M| ID LSB|
             - - +-+-+-+-+-+-+-+-+

                 0               7
             - - +-+-+-+-+-+-+-+-+ - -
    B5           |1 0 1|M|H D|T|1|
             - - +-+-+-+-+-+-+-+-+ - -

                                                1               2
                 0               7 8            5               3
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    B6           |1 1 0|M|t|          TS*       |     ID LSB    |
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                                1
                 0               7 8            5
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
    B7           |1 1 1|M|C|H|S|D|T|I|t|   SEQ*  |
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -

                                                1
                 0               7 8            5
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    C1, D1       |0 0 1|M|          SEQ*          |
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

                                                1
                 0               7 8            5
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
    C2, D2       |0 1 0|M|C|H|S|      SEQ*       |
             - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
```

```
                 0             7
          - - +-+-+-+-+-+-+-+-+ - -
C3, D3        |0 1 1|M|C|H|S|-|
          - - +-+-+-+-+-+-+-+-+ - -

                 0             7 8             5
          - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
D5            |1 0 1|M|       ID LSB        |
          - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
```



```
                                            1 1             2
                 0             7 8          5 6             3
          - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
D6            |1 1 0|M|C|H|S|-|             ID              |
          - - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
```

Bit masks indicating additional fields have the following meaning:

C - Traffic (C)lass / Type Of Service
H - (H)op Limit / Time To Live
S - Contributing (S)ources - CSRC
D - Timestamp (D)elta
T - (T)imestamp LSB
I - (I)dentification LSB

If any of these fields are included, they will appear in the order as listed above and the format of the fields will be as described below.

C - Traffic Class / Type Of Service

The field contains the value of the original IP header field.

```
            8 bits
   - - +-+-+-+-+-+-+-+-+ - -
       |   TC / TOS    |
   - - +-+-+-+-+-+-+-+-+ - -
```

H - Hop Limit / Time To Live

The field contains the value of the original IP header field.

```
            8 bits
   - - +-+-+-+-+-+-+-+-+ - -
       |   HL / TTL    |
   - - +-+-+-+-+-+-+-+-+ - -
```

S - Contributing Sources

The CSRC field is built up of:

```
- a counter of the number of CSRC items present (4 bits)
- an unused field (4 bits)
- the CSRC items, 1 to 15 (4-60 octets)

      1 octet       +      4 to 60 octets
- - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - -
    | Count | Unused|       CSRC Items        |
- - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+- - -
```


D - Timestamp Delta

The Timestamp Delta field is a one-octet field. We want to communicate Timestamp Delta values corresponding to 80 ms. Therefore, the Timestamp Delta value communicated is not the actual number of samples, but the number of samples divided by 8. Thus, only Timestamp Delta values evenly divisible by 8 can be communicated in the Timestamp Delta field of an extension. On the other hand, the maximum value is 255*8 = 2040 (255 ms at 8000 Hz). Delta values larger than 2040 or delta values not evenly divisible by 8 must be communicated in a DYNAMIC packet.

```
             8 bits
- - +-+-+-+-+-+-+-+-+ - -
    |Timestamp Delta|
- - +-+-+-+-+-+-+-+-+ - -
```

Note that when the Timestamp Delta is changed, Timestamp LSB field MUST also be included not downscaled with the delta.

T - Timestamp LSB

The field contains the 16 least significant bits of the RTP timestamp, scaled if r-bit not set. May be sent as extended range for some profiles.

```
                  16 bits
- - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
    |              TS*              |
- - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ - -
```

I - Identification

The field contains the IP Identification.

```
                  16 bits
- - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |              ID               |
- - +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

When information of any kind is sent in an extension, the

corresponding information SHOULD also be sent in some subsequent packets (either as Extensions or in DYNAMIC packets).

5.7.5. Feedback packets

Feedback packets are used by the decompressor to provide various types of feedback to the compressor. That could include active feedback to assure error free performance or passive feedback (in case of invalidated context) to request a context update from the compressor. The feedback mechanisms defined here leave a lot to the implementation regarding how to use feedback. The general feedback packet format is shown below:

```
                          0   1   2   3   4   5   6   7
                         +...+...+...+...+...+...+...+...+
  FEEDBACK (GENERAL)     :  Context Identifier  (CID)    :
                         +---+---+---+---+---+---+---+---+
                         | 1   1   1   1   0 |    Type   |
                         +---+---+---+---+---+---+---+---+
```

Note that The CID field is present only for profiles using STATIC packet format 2 or 4, which are profiles supporting multiple packet streams. The Type field tells what kind of feedback the packet corresponds to and the feedback types defined are the following:

```
                          0   1   2   3   4   5   6   7
                         +...+...+...+...+...+...+...+...+
  STATIC_FAILURE_FEEDBACK :  Context Identifier  (CID)   :
                         +---+---+---+---+---+---+---+---+
                         | 1   1   1   1   0 | 0   0   0 |
                         +---+---+---+---+---+---+---+---+
```

The STATIC_FAILURE_FEEDBACK packet tells the compressor that the static part of the decompressor context is invalid, and that an update of that part is required. Reasons for sending such feedback could be that no STATIC packet has been received at all, or that decompression has failed even when DYNAMIC packets are decompressed.

```
                          0   1   2   3   4   5   6   7
                         +...+...+...+...+...+...+...+...+
  INVALID_CONTEXT_FEEDBACK :  Context Identifier  (CID)  :
                         +---+---+---+---+---+---+---+---+
                         | 1   1   1   1   0 | 0   0   1 |
                         +---+---+---+---+---+---+---+---+
                         |  Last Sequence Number LSB     |
                         +---+---+---+---+---+---+---+---+
```

The INVALID_CONTEXT_FEEDBACK packet SHOULD be sent to signal an invalid decompressor context, indicated by failing decompression of COMPRESSED packets.

```
                             0   1   2   3   4   5   6   7
                           +...+...+...+...+...+...+...+...+
   NO_PACKETS_FEEDBACK     :    Context Identifier  (CID)   :
                           +---+---+---+---+---+---+---+---+
                           | 1   1   1   1   0 | 0   1   0 |
                           +---+---+---+---+---+---+---+---+
                           |    Last Sequence Number LSB    |
                           +---+---+---+---+---+---+---+---+
```

The NO_PACKET_FEEDBACK packet can be used by the decompresser to signal that packets have not been received for some time. It is not always possible for the decompressor to notice such events, and it is therefore up to the implementers to decide whether and when to use this feedback packet.

```
                             0   1   2   3   4   5   6   7
                           +...+...+...+...+...+...+...+...+
   LONGEST_LOSS_FEEDBACK   :    Context Identifier  (CID)   :
                           +---+---+---+---+---+---+---+---+
                           | 1   1   1   1   0 | 0   1   1 |
                           +---+---+---+---+---+---+---+---+
                           |    Last Sequence Number LSB    |
                           +---+---+---+---+---+---+---+---+
                           |     Length of longest loss     |
                           +---+---+---+---+---+---+---+---+
```

The LONGEST_LOSS_FEEDBACK packet can be used by the decompressor to inform the compressor about the length of the longest loss event that has occurred on the link between compressor and decompressor. It is not always possible for the decompressor to provide this information, and it is therefore up to the implementers to decide whether and when to use this feedback packet.

```
                             0   1   2   3   4   5   6   7
                           +...+...+...+...+...+...+...+...+
  CONTEXT_UPDATED_FEEDBACK :    Context Identifier  (CID)   :
                           +---+---+---+---+---+---+---+---+
                           | 1   1   1   1   0 | 1   0   0 |
                           +---+---+---+---+---+---+---+---+
                           |    Last Sequence Number LSB    |
                           +---+---+---+---+---+---+---+---+
```

The CONTEXT_UPDATED_FEEDBACK packet can be used to signal that an update of some header fields has been correctly received, either in a DYNAMIC packet or in an EXTENDED_COMPRESSED packet. It is optional to use this active feedback mechanism and the compressor MUST NOT expect such packets initially. First after reception of one such packet, the compressor can expect to get this feedback from the decompressor.

5.8. Encoding of field values

The source increases the RTP sequence number by one for each packet sent. However, due to losses and reordering before the compression point, the changes seen by the compressor may vary. This would especially be the case if we consider the scenario in Figure 1.1 where there are cellular links at both ends of the path. That is one reason why sequence number changes need special treatment, but another reason is that both timestamps and IP identification for most packets can be recreated with a combination of history and sequence number knowledge. The profiles defined in this document handle the sequence number, timestamp and identification values with LSB encoding, except for some profiles that use LSP encoding for the sequence number. For timestamp, some profiles also use the principle with timer-based decompression. This chapter describes how the different encoding methods are applied to the various field values.

5.8.1. LSB encoding of field values

LSB encoding is used for sequence number, timestamp and identification encoding as described in chapter 4.5.1. The sequence numbers, included in all compressed headers, can be sent with extended range in extension headers. This is also the case with the timestamp value when not using timer-based TS reconstruction (see 5.7 and 5.7.8). With timer-based timestamp decompression, the amount of timestamp LSB that is sent is always limited to the size of the field in the compressed header. Note that in most headers, the timestamp value is sent as STS LSB (scaled timestamp LSB), which means that it is the least significant bits of the timestamp, scaled down with the timestamp delta (STS LSB = LSB of (TS / TS Delta)).

5.8.2. LSP encoding of field values

LSP, as described in chapter 4.5.2, is used for sequence numbers in the "Sequence LSP" field of COMPRESSED1..COMPRESSED4 headers. For those headers, there are 28 code-points left for sequence information because 4 are reserved for packet type identification. An LSP of size 28 is therefore used with the following encoding:

CODE(n) = LSP:28(n)

The sequence range can be extended with extra bits in extension headers, as described in chapter 4.5.3. The "SEQ EXT LSB" field must for the case of extended LSP consist of the LSB of the integer quotient.

The reordering parameter for LSP MUST be set to 2 meaning that first and second order reordering can be handled by the encoding.

5.8.3. Timer-based timestamp decompression

The RTP timestamp field is one of the header fields that may change dynamically on a per packet basis. For audio services, the timestamp value can be inferred from the encoded RTP sequence number value during talk spurts. When the encoded sequence number is incremented by N, the timestamp value is incremented by N * Timestamp-Delta-value. However, when a talk spurt has faded into silence and a new talk spurt starts, the timestamp value will take a leap compared to the sequence number. To communicate this leap in the timestamp value, some additional action has to be taken. In chapter 5.7.4, extension headers are defined that can transfer this leap in the timestamp value. That increases, however, the average header size. This chapter describes an optional method used by some profiles (see the TbT column of table 5.1) to reconstruct the timestamp value, requiring only a fixed number of added bits for timestamp leaps. The method makes use of timers or a local wall clock at the decompressor.

To initialize the header compression and the timer-based timestamp reconstruction, the absolute value of the timestamp together with the sequence number must be transferred from compressor to decompressor at the beginning of the compression session. A default timestamp delta is also transferred. This is done through the transmission of a DYNAMIC header. For speech codecs with 8 kHz sampling frequency and 20 ms frame sizes, for example, the timestamp delta will be 8000*0.02 = 160. The decompressor then knows that the timestamp will increase by 160 for each packet containing 20 ms of speech. Hence, by using a local clock and by measuring packet arrival times, the decompressor can estimate the timestamp change compared to the previous packet. If, for example, a speech period has been succeeded by a silence period at the time T0 and a new speech period starts at the time T0+dT, it can be assumed that the timestamp has changed by:

    round(dT/(time for one speech frame)) * (timestamp delta)

The packet time interval (or codec frame size in time) may be determined through the a priori knowledge that most speech codecs have constant frame sizes of 10, 20 or 30 ms, or through measurements on packet arrival times.

The decompressor can then get an estimate of the timestamp change, add this change to the previous value and replace the least significant bits with those received in the compressed header. This should give the correct timestamp value.

It is very important to verify the correctness of a timer-based timestamp decompression. However, this is automatically done in ROCCO with the header compression CRC verification.

5.9. Header compression CRCs, coverage and polynomials

This chapter contains a description of how to calculate the different CRCs used in the packet headers defined in this document.

5.9.1. STATIC packet CRC

The CRC in the STATIC header is calculated over the whole STATIC packet except for the header compression CRC itself. Therefore, the header compression CRC field MUST be set to 0 before the CRC calculation.

The CRC polynomial to be used in STATIC packets is:

$$C(x) = 1 + x + x^2 + x^8$$

5.9.2. DYNAMIC packet CRC

The CRC in the DYNAMIC packet is calculated over the original IP/UDP/RTP header. Before the calculation of the CRC, the IPv4 header checksum and the UDP checksum have to be set to 0. This makes it possible to recalculate the checksums after the decompression. Calculation over the full IP/UDP/RTP headers ensures that the decompressed IP/UDP/RTP header is a correct header.

The CRC polynomial to be used in DYNAMIC packets is:

$$C(x) = 1 + x + x^2 + x^8$$

5.9.3. COMPRESSED packet CRCs

COMPRESSED1..COMPRESSED4

The header compression CRC in COMPRESSED header types 1 to 4 is calculated over the same headers as the CRC in the DYNAMIC packet, except for profiles which use replacement of the UDP checksum, i.e. except for profiles 1-4 and 13-16. In profiles 1-4 and 13-16, the header compression CRC also covers the payload covered by the UDP checksum.

The polynomial to be used is:

$$C(x) = 1 + x + x^4 + x^5 + x^9 + x^10$$

COMPRESSED5..COMPRESSED8 and COMPRESSED13..COMPRESSED16

In COMPRESSED header types 5 to 8 and 13 to 16 the header compression CRC is calculated over the same headers as the CRC in the DYNAMIC packet, but with a different polynomial:

$$C(x) = 1 + x + x^3$$

COMPRESSED9..COMPRESSED12

In COMPRESSED header types 9 to 12 the header compression CRC is calculated over the same headers as the CRC in the DYNAMIC packet, but with a different polynomial:

C(x) = 1 + x + x^3 + x^4 + x^6

COMPRESSED17..COMPRESSED20

In COMPRESSED header types 17 to 20 the header compression CRC is calculated over the same headers as the CRC in the DYNAMIC packet, but with a different polynomial:

C(x) = ????

5.10 State transitions using keyword packets

(Editor: This section is separate because I believe merging it in is not just an editorial exercise.)

5.10.1 The concept of keyword based state transitions

The main purpose of this algorithm is to enable the compressor to employ the optimistic approach (for uni- or bi-directional links) as described in greater detail in sections 5.3.1 and 5.3.2.

For an optimistic approach the compressor decides when it wants to proceed from FO to SO state. Basically the transition will be made after the compressor thinks the FO packets are received correctly and the valid context is established. However this is an optimistic and not a reliable approach. The compressor might proceed to SO state and send SO packets, while the decompressor lost packets and is still in FO state. Therefore a mechanism is needed to detect this case.

This section describes in greater detail the mechanism of using keyword packets to transit securely from FO to SO state.

5.10.1.1 Keyword field, update and non-update packet

The algorithm is based on the concept that some packets update the context, namely update packets, while others do not update the

context at all, namely non-update packets. All packets indicate the context state that is referenced and therefore needed to decompress the packet correctly.

The main idea how this is done is the definition of a keyword field. The packets with the same keyword field value, reference the same context state. The context state to be used is defined by sending a update packet, i.e. a packet that has a new keyword value and which contents update the context to the new context state. The following packets are called non-update packets, because they do not update the context.

Hence, if a non-update packet gets lost, the receiver is nevertheless able to decompress the following packets.

5.10.1.2 Refreshing the context by sending update packets

From time to time it will be necessary to update the context. There

are mainly two reasons to do so.

First, while compressing and transmitting the compressed non-update packets, the LSB encoded values may increase and need more coded bits in the compressed header. If the header size exceeds a certain threshold, a new keyword should be sent in an update packet. This enables the compressor to use less LSB in the following non-update packets. E.g. after a while the number of LSB to encode the RTP sequence number will grow. If this value exceeds 6 bits, it might be useful to send an update packet, because the information would not fit into an one-byte header packet any more. After the successful update the compressor is able to send one byte header packets again.

This means that the compressor is still in SO state and thus sends SO packets. The corresponding update packets that are sent are also SO packets, because they still rely on the previous update. This also means that the update packets are small, i.e. only two bytes in size.

Second, if a value had changed and seems to be stable now, a new update packet should be sent. This means a transition from SO to FO state happened. It is not possible to use SO packets any more, because some fields can not be calculated from the RTP sequence number any more. It is not necessary to send update packets in FO state. The problem would be, that changed value would have to be transmitted in all following packets. This means that FO packets have to be send all the time, i.e. the compressor stays in FO state. To get into SO state, the compressor has to use the optimistic approach, which says that if he thinks the decompressor has established the new context he switches to SO state. To update the context at the decompressor, update packets have to be sent.

E.g. after a silent period the timestamp changes by another value than the default difference timestamp. From this on it is not

possible for the decompressor to calculate the timestamp from the RTP sequence number. The compressor is in FO state and either sends the LSB of the timestamp in every packet, or updates the context, to enable a later transition to SO state again.

5.10.1.3 Minimizing the loss probability of update packets (safe transition from FO to SO state)

This section describe the transition from FO to SO state in greater detail and gives algorithms that support a save transition.

It is useful to send several update packets with the same keyword value to establish a new context state for both sides, before going to SO state. The LSB encoded values are transmitted as usual in those packets, while one has to take care that the original values of the fields that changed irregularly are transmitted in every of those update packets. The use of this mechanism reduces the context state loss probability, because only one of those update packets has to be received correctly.

Sending several update packets with the same keyword could be done either successively or in any kind of sparse mode, e.g. as described in [ref to sparse mode description, TBD].

5.10.1.4 Restrict the use of new keywords

The number of different keywords is limited by the number of bits used for the keyword field. Here only one bit is used, which leads to two different keywords. To ensure that consecutive packet loss of a few packets does not lead to wrong decompression, the use of new keyword values must be limited.

It is only allowed to send a new keyword in an update packet, if N non-update packets were sent since the last keyword change. The value N should be set according to the expected longest loss event. This restriction is possible, because one never is forced to send an update packet. It is always possible to send all information in a non-update packet. This might lead to a decreased efficiency for short times, because the compressor stays longer in FO state, but if the keywords are used properly, this should only very seldom happen.

To use the keyword properly means that it is only changed if the compressor is rather sure that the values will remain constant for the next packets. An example of a non-properly used keyword change is the definition of a new default delta timestamp value (in an update packet), just because it changed for one packet. This might be due to a silent period and might change back to the original value in the next packet again.

If the compressor follows this restriction, more than N consecutive packets have to be lost, before the decompressor would not detect the

loss of the update packet. To avoid even this situation a time-out might be applied, after which the decompressor will only accept new update packets or Full Header packets.

5.10.1.5 Loss of update packets

Only if the update packets are transmitted correctly, the receiver is able to decompress any incoming compressed header (i.e. the receiver is then in SO state). Therefore if the update packets are transmitted multiple times, the probability that none of this packets is received, is very low. However, packet loss may occur while transmitting update packets. In case none of the update packets was received and the decompressor received a packet with a new keyword that is not an update packet, it must send a message to the compressor, to ask for a packet with a header that re-establishes its context. This is always an update packet or a Full Header packet.

5.10.1.6 The use of LSB encoding in the context of this algorithm

The packets that follow an update packet, are encoded by transmitting the Least Significant Bits (LSB) of regular changing fields (e.g. RTP Sequence Number). In some cases it might not be necessary to transmit the regular changing fields at all, e.g. if the timestamp can be calculated from the sequence number it is not transmitted. The packets also contain the original values of fields that did change since the last update, but are usually assumed to be constant (e.g. RTP Marker bit, RTP Payload Type).

A problem in using LSBs is the wrap-around. Because only some bits of the original value is transmitted, it has to be ensured, that the decompression is correct. If other bits than the transmitted bits have changed, the decompressor must be able to compute this.

To solve this problem Variable Length Encoding (VLE) as described in [ref to VLE] is used.

5.10.1.7 Adaptation to the environment

The compressor has a lot of freedom in the compression algorithm. Even though the use of new keywords is restricted, the compressor decides when the keywords should be changed. Two strategies are possible, which are a trade-off between compression efficiency and robustness against packet loss. One possibility is to send a new keyword as often as needed and possible. E.g. the keyword is changed, if the header size exceeds 1 byte. Another possibility is to send new keywords less frequent. While on the one hand the compression efficiency might be better in the first case, the second possibility is more robust and less susceptible for packet loss.

Using this freedom the compressor may adapt the compression to the environment (i.e. the experienced BER or RTT). Another parameter of

the environment that should be taken into consideration is the assignment of the IPv4 Identification value. While it is possible to have a totally random IP Identification, it might also be possible that it is increased for every packet by a fixed value (sequential IP ID). Different sets of packet types, used for different environments might lead to a better performance. This paper defines two different packet type sets. The packet formats are optimised for different environments. If the IP ID is assigned sequentially, increasing by a fixed value for each packet, the header compression mechanism should take advantage of it. Anyway, because we cannot assume this behaviour, another set of packet formats is defined, which is optimised for non sequential IPv4 Identification values.

The two sets of packet formats are called packet-profiles in the remainder of this document.

5.10.1.8 Dealing with Residual Bit Error Rate

A requirement from the lower layers that this header compression scheme works above, is that the residual bit error rate should be kept to a minimum. However bit errors might occur in compressed packet. To avoid a damage propagation (see [requirements document] for the definition of damage propagation) the update packets are protected by a CRC, which is calculated over the uncompressed header. Detailed information about the CRC and its usage can be found in section [CRC usage]. Because only the update packets update the context on which the non-update packets rely, damage propagation is prevented, by protecting only the update packets.

5.10.3 Packet-Profile 1, optimized for sequential IPv4 Identification

This section shows five different packets that are used to transmit the data and signal errors from the receiver to the sender. The

packet formats are optimised for the use in an environment, where the IPv4 Identification is assigned sequentially for the compressed packet stream. The format of the packets is described and it is explained in detail how the decompressor is able to regenerate the complete header from the given information. The exact compression behaviour is implementation specific, but it has to be ensured, that any decompressor is able to regenerate the complete header in the described way.

5.10.3.1 Full Header packet

The Full Header packet is sent at the beginning of the session to establish a valid context, i.e. to switch from Init- to FO state. It is only sent another time if requested by the receiver or a severe error occurred. The receiver must request a Full Header packet only if the initial packet was lost, or a severe error occurred, that cannot be solved by a Compressed Packet.

To ensure the correct reception of the fields that are only transmitted in this packet type, it might be useful to use this packet type for several succeeding packets. The next packet type to use is always an update packet, which contains a new keyword. The decompressor will discard any other received packet and sent a context state feedback, until it receives an update packet to establish a valid context (the keyword is part of the context).

The format of this packet's header is quite similar to the original IP/UDP/RTP header. However, as described in other header compression papers, such as CRTP [7], the length fields of the IP and UDP packets are redundant. They are usually signalled by the link layer. This enables us to use these fields to signal the header compression specific session context identifier (CID)as follows:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|C-L|                           |   First length field
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|     (CID)     |     (CID)     |   Second length field
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

C-L (CID Length):
  00 - no CID
  01 - 8 bit
  10 - 16 bit

The selection of 0, 8 or 16 bit CIDs enables the compressor to set-up enough sessions while keeping the overhead to a minimum.

Packet type identification might not be done by the link layer. In this case another byte is added before the original full header:

```
+-+-+-+-+-+-+-+-+
|1|1|1|0|-|-|-|-|    Packet Type Identification
```

```
+-+-+-+-+-+-+-+-+
:  RTP/UDP/IP   :
:    packet     :
```

5.10.3.2 Basic-Compressed packet

FO packets are always of this type. Only if no extensions are transmitted, this is an SO packet. This is useful either to enlarge the number of RTP sequence number bits, or to send an update packet out of SO state.

The header of the Basic-Compressed packet is divided into a basic header that is transmitted for every packet of this type and header

extensions that are only used if necessary. Update and non-update packets can be sent in Basic-Compressed packet format. The type is identified by the new keyword flag, which is set for update packets.

5.10.3.2.1 Basic header

The basic header's format is as follows:

```
  0   1   2   3   4   5   6   7
+...+...+...+...+...+...+...+...+
:       MSB of Session CID      :  if 16 bit CID is used
+...+...+...+...+...+...+...+...+
:       LSB of Session CID      :  if 16 or 8 bit CID is used
+---+---+---+---+---+---+---+---+
| 1 | 1 | C |KW |NKW| M | E | S |
+---+---+---+---+---+---+---+---+
:          LSB of RTP SN        :
+...+...+...+...+...+...+...+...+
:          MSB of RTP SN        :  if S==1
+...+...+...+...+...+...+...+...+
:                               :
:          Extension(s)         :  if E==1
:                               :
+...+...+...+...+...+...+...+...+
:                               :
+          UDP Checksum         +  if non-zero in context
:                               :
+...+...+...+...+...+...+...+...+
:              CRC              :
+---+---+---+---+---+---+---+---+
|           RTP Data            |
:                               :
```

CID:
The first two bytes can be used for the session CIDs.

KW (Keyword):
The Keyword field must be present in every packet. To detect loss of update packets, it must be changed at each renewal.

NKW (New Keyword Indication):
If this bit is set, the compressor defines this packet as an update packet. The context state after decompressing this packet is stored and referenced in the following packets. Several successive update packets should be sent, each containing the relevant information, to

ensure the reception at the decompressor. This bit also indicates the presence of the CRC.

M (RTP Marker):
The M-bit represents the original RTP Marker.
E (Extension):
This bit indicates that at least one extension is used. The different possible extensions, that are used to transmit information about

irregular changes in the header fields, are described in detail in the following sections.

S (RTP Sequence Number Indication):
This bit indicates if the LSB of the RTP Sequence Number or the original value follows directly.
  S=0: 8 bit LSB RTP Sequence Number
  S=1: 16 bit RTP Sequence Number

UDP Checksum:
If the UDP Checksum is enabled, this field contains the 16-bit Checksum, else it is not present.

CRC:
This 8 bit CRC is calculated over the uncompressed header. It is used to verify the correct transmission of the compressed packet.

5.10.3.2.6 Changing Field Extension

This extension is sent every time some header fields changed in an irregular way and cannot be calculated from the RTP Sequence Number. This might be the case e.g. for the RTP Timestamp after a silent period, or for the IPv4 Time To Live value. If the MTW-bit is set (i.e. the packet is an update packet), the fields transmitted in this extension define the new context state to be referenced by the following packets. Several successive update packets should be sent, each containing the relevant fields, to ensure the reception at the decompressor.

The format of the Changing Field Extension is defined below:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 0 |  ID   |TS |TOS|TTL|PT | E |
+---+---+---+---+---+---+---+---+
:                               :
:   (LSB) IPv4 Identification   :  if ID > 0
:                               :
+...+...+...+...+...+...+...+...+
:                               :
:       (LSB) RTP Timestamp     :  if TS == 1
:                               :
+...+...+...+...+...+...+...+...+
:      IPv4 Type of Service     :  if TOS == 1
+...+...+...+...+...+...+...+...+
:      IPv4 Time To Live        :  if TTL == 1
+...+...+...+...+...+...+...+...+
:      RTP Payload Type     : ~ :  if PT == 1
```

```
+...+...+...+...+...+...+...+...+
```

The first bit (0) indicates the extension that is used.

ID (IPv4 Identification Indication):
This bit indicates if the original IPv4 Indication value is transmitted in the IPv4 Identification field or the LSB of the Identification or nothing.

ID=0: No IPv4 Identification
ID=1: 8 bit LSB IPv4 Identification
ID=2: 16 bit IPv4 Identification
ID=3: not used

TS (RTP Timestamp Indication):
This bit indicates if the RTP Timestamp is transmitted. If it is set to one, one of the following fields are used in the (LSB) RTP Timestamp field:

```
+---+---+---+---+---+---+---+---+
| 0 |                           |
+---+  15 LSB of RTP Timestamp  +
|                               |
+---+---+---+---+---+---+---+---+

+---+---+---+---+---+---+---+---+
| 1 | 0 |                       |
+---+---+                       +
|                               |
+       22 LSB of RTP Timestamp +
|                               |
+---+---+---+---+---+---+---+---+

+---+---+---+---+---+---+---+---+
| 1 | 1 | 0 |                   |
+---+---+---+                   +
|                               |
+       29 LSB of RTP Timestamp +
|                               |
+                               +
|                               |
+---+---+---+---+---+---+---+---+

+---+---+---+---+---+---+---+---+
| 1 | 1 | 1 | 0 | - - - -       |
+---+---+---+---+---+---+---+---+
|                               |
+                               +
|                               |
+          RTP Timestamp        +
|                               |
+                               +
|                               |
+---+---+---+---+---+---+---+---+
```

TOS (IPv4 Type Of Service Indication):

This bit indicates the transmission of the IPv4 Type Of Service value in the IPv4 Type Of Service field.

TTL (IPv4 Time To Live Indication):
This bit indicates the transmission of the IPv4 Time To Live value in the IPv4 Time To Live field.

PT (RTP Payload Type Indication):
This bit indicates the transmission of the RTP Payload Type value in the RTP Payload Type field.

E (Extension):
This bit indicates that another extension follows this one.

5.10.3.2.3 Default Delta Extension

The compressor will follow the changes in the RTP Timestamp values and the IPv4 Identification values, relative to the changes in the RTP Sequence Number values. To do this a delta value according to the following equation might be used:

```
dTS = (TS(n)-TS(n-1)) / (SN(n)-SN(n-1)), with

    dTS      : delta Timestamp
    TS(n)    : Timestamp of current packet
    TS(n-1)  : Timestamp of previous packet
    SN(n)    : Sequence Number of current packet
    SN(n-1)  : Sequence Number of previous packet
```

If the compressor detects that for several packets the delta timestamp or delta identification value is the same, this delta value can be used to calculate the timestamp or identification from the sequence number. To do so, the decompressor has to be informed about this default delta value. The compressor uses this extension to signal default delta timestamp (ddTS) or default delta identification (ddID) values to the decompressor. This extension should be sent in update packets only. If it is used, a change extension, containing the timestamp or respectively the identification field must be sent as well.

The format of the Default Delta Extension is given below:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
| 1 | 0 | - | - | ddTL  | ddIDL |
+---+---+---+---+---+---+---+---+
:             ddTS              :   if ddTL > 0
+...+...+...+...+...+...+...+...+
:             ddTS              :   if ddTL > 1
+...+...+...+...+...+...+...+...+
:             ddTS              :   if ddTL > 2
```

```
+---+---+---+---+---+---+---+---+
|             ddID              |   if ddIDL > 0
+---+---+---+---+---+---+---+---+
|             ddID              |   if ddIDL > 1
+---+---+---+---+---+---+---+---+
```

The first four bits identify this extension.

ddTL (default delta Timestamp Length):
This field indicates the length of the default delta Timestamp field:
- ddTL=0: no default delta Timestamp field present
- ddTL=1: 1 byte
- ddTL=2: 2 byte
- ddTL=3: 3 byte

ddIDL (default delta Identification Length):
This field indicates the length of the default delta Identification field:
- ddIDL=0: no default delta Identification field present
- ddIDL=1: 1 byte
- ddIDL=2: 2 byte
- ddIDL=3: not used

5.10.3.3 One-Byte-Header or Two-Byte-Header packet

Packets of these two types are always non-update packets. Since they only contain parts of the RTP sequence number they can only be sent in SO state and therefore they are SO packets. They reference the last update packet and carry the same keyword value.

If the compressor communicated the default delta values to the decompressor and all changes are regular, the decompressor should be able to recalculate the identification and timestamp value from the sequence number. Hence it is not necessary to transmit these values in all packets.

The One-Byte-Header or Two-Bytes-Header packets cannot be used if other fields than the sequence number, timestamp and identification changed. The timestamp and identification also have to change according to the following equations:

```
TS(n) = TS(n-1) + (SN(n) - SN(n-1)) * ddTS
ID(n) = ID(n-1) + (SN(n) - SN(n-1)) * ddID

   ddTS     : default delta Timestamp
   ddID     : default delta Identification
   TS(n)    : Timestamp of current packet
   TS(n-1)  : Timestamp of previous packet
   SN(n)    : Sequence Number of current packet
   SN(n-1)  : Sequence Number of previous packet
```



```
   ID(n)    : Identification of current packet
   ID(n-1)  : Identification of previous packet
```

In this state the compressor might use the One-Byte-Header or Two-

Byte-Header packet. These packets contain only the LSB of the RTP Sequence Number and the keyword, which is enough information for the decompressor to regenerate the original header.

The packet format for the One-Byte-Header packet is given below:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
:       MSB of Session CID      :  if 16 bit CID is used
+---+---+---+---+---+---+---+---+
:       LSB of Session CID      :  if 16 or 8 bit CID is used
+---+---+---+---+---+---+---+---+
| 0 |KW |LSB RTP Sequence Number|
+---+---+---+---+---+---+---+---+
```

The packet format for the Two-Byte-Header packet is given below:

```
  0   1   2   3   4   5   6   7
+---+---+---+---+---+---+---+---+
:       MSB of Session CID      :  if 16 bit CID is used
+---+---+---+---+---+---+---+---+
:       LSB of Session CID      :  if 16 or 8 bit CID is used
+---+---+---+---+---+---+---+---+
| 1 | 0 |KW |                   |
+---+---+---+                   +
|       LSB RTP Sequence Number |
+---+---+---+---+---+---+---+---+
```

The decision which of these packets is to be used should be done according the context RTP sequence number. The not transmitted MSB of the RTP sequence number must not change.

5.10.3.4 Context State packet

This header compression mechanism is aimed to perform good, even if used over an unreliable channel. Hence bit errors can occur quite frequently and packets will get lost. If the lost packet was a non-update packet, this does not effect the decompressor at all, but reception of a non-update packet with a new keyword, without receiving an corresponding update packet invalidates the decompressor's context. From this moment on any compressed packet, even if it was received correctly, cannot be decompressed, until the context is updated correctly again.

To minimize the probability of this situation, several successive update packets should be sent (with the same keyword). But even all

of these packets might get lost. Hence a mechanism is needed to inform the compressor about a lost context, to request an update packet.

To request a context update, the decompressor must send immediately after detecting an invalid context, a Context State packet. This packet contains the last correctly received keyword and RTP Sequence Number. The compressor knows at reception of such a Context State packet, what information it has to send in the update extension, to

update the decompressor's context correctly.

Another error, that could occur, is the loss of the first packet, i.e. the Full Header packet. Since most of the header information, e.g. addresses and ports, are transmitted only in this packet, it is essential for the decompressor to establish a valid context to receive this packet. If the decompressor receives a Compressed packet, with a new session CID, that was not initialized, by a Full Header packet, this Full Header packet must have been lost. In this case the decompressor must request a new Full Header packet, by the means of the Context State packet.

The format of the Context State packet is as follows:

```
+---+---+---+---+---+---+---+---+
:       MSB of Session CID      :  if 16 bit CID is used
+---+---+---+---+---+---+---+---+
:       LSB of Session CID      :  if 8 or 12 bit CID is used
+---+---+---+---+---+---+---+---+
|FHL|KW |                       |
+---+---+---+---+---+---+---+---+
:                               :
+       RTP Sequence Number     +  if FHL == 0
:                               :
+---+---+---+---+---+---+---+---+
```

CID:
The first two bytes can be used for the session CID.

FHL (Full Header Lost):
If this bit is set to one, the first Full Header packet was lost, no context was established and a new Full Header packet is requested. If it is set to zero a context update is required and the RTP Sequence Number of the last correctly decompressed packet is transmitted as well.

5.10.4 Packet-Profile 2, optimized for non-sequential IPv4 identification

This section shows five different packets that are used to transmit the data from the sender to the receiver and signal errors from the

receiver to the sender. The packet formats are optimised for the use in an environment, where the IPv4 Identification is not assigned strictly sequentially for the compressed packet stream. The identification value is expected to increase by a small random number (e.g. smaller than 64). The format of the packets is described and it is explained in detail how the decompressor is able to regenerate the complete header from the given information. The exact compression behaviour is implementation specific, but it has to be ensured, that any decompressor is able to regenerate the complete header in the described way.

5.10.4.1 Full Header packet

The Full Header packet is sent at the beginning of the session to

establish a valid context, i.e. to transit from Init- to FO state. It is used exactly as in packet-profile 1 and has the same format (see section 5.10.3.1 for details).

5.10.4.2 Basic-Compressed packet

The header of the Basic-Compressed packet is divided into a basic header that is transmitted for every packet of this type and header extensions that are only used if necessary. As described for the previous packet-profile, this packet can be used for update packets (new-keyword flag set to one) or non-update packets. As described before if no extensions are used, this packet can be sent in SO state and therefore actually is an SO packet. Else it is an FO packet.

5.10.4.2.1 Basic header

The basic header's format is as follows:

```
  0   1   2   3   4   5   6   7
+...+...+...+...+...+...+...+...+
:       MSB of Session CID      :  if 16 bit CID is used
+...+...+...+...+...+...+...+...+
:       LSB of Session CID      :  if 16 or 8 bit CID is used
+---+---+---+---+---+---+---+---+
| 1 | 1 | 0 |KW |NKW| M | E |S/I|
+---+---+---+---+---+---+---+---+
: Type  :                       :
+...+...+  Sequence Number &    +  if S/I==1
:          Identification       :
+...+...+...+...+...+...+...+...+
:                               :
:          Extension(s)         :  if E==1
:                               :
+...+...+...+...+...+...+...+...+
:                               :
+          UDP Checksum         +  if non-zero in context
:                               :
```



```
+...+...+...+...+...+...+...+...+
:              CRC              :
+---+---+---+---+---+---+---+---+
|           RTP Data            |
:                               :
```

CID:
The first two bytes can be used for the session CIDs.

KW (Keyword):
The Keyword field must be present in every packet. To detect loss of update packets, it must be changed at each update.

NKW (New Keyword Indication):
If this bit is set, the compressor defines this packet as an update packet. The context state after decompressing this packet is stored and referenced in the following packets. Several successive update packets should be sent, each containing the relevant information, to ensure the reception at the decompressor. This bit also indicates the

presence of the CRC.

M (RTP Marker):
The M-bit represents the original RTP Marker.

E (Extension):
This bit indicates that at least one extension is used. The different possible extensions, that are used to transmit information about irregular changes in the header fields, are described in detail in the following sections.

S/I (RTP Sequence Number & Identification Indication):
This bit signals that the LSB of the RTP Sequence Number and IPv4 Identification follow directly.

Type:
These two bits indicate how the following bytes are used for the Sequence Number and Identification:

```
Type = 0:
  7 bit RTP Sequence Number
  7 bit IPv4 Identification
Type = 1:
  6 bit RTP Sequence Number
  16 bit IPv4 Identification
Type = 2:
  8 bit RTP Sequence Number
  14 bit IPv4 Identification
Type = 3:
  10 bit RTP Sequence Number
  12 bit IPv4 Identification
```

UDP Checksum:

If the UDP Checksum is enabled, this field contains the 16-bit Checksum, else it is not present.

CRC:
This 8 bit CRC is calculated over the uncompressed header. It is used to verify the correct transmission of the compressed packet.

5.10.4.2.2 Changing Field Extension

This extension is used similar as in packet-profile one and also has the same format. For details see section 5.10.3.2.2.

5.10.4.2.3 Default Delta Extension

This extension is used similar as in packet-profile one and also has the same format. For details see section 5.10.3.2.3.

5.10.4.2.4 Two-Byte-Header or Three-Byte-Header packet

These two packet types can only be used for non-update packets. They reference the last update packet and therefore carry the same keyword value. These packets can only be sent in SO state and therefore are SO packets.

```
If the compressor communicated the default delta values to the
decompressor, the decompressor should be able to recalculate the
timestamp value from the sequence number. Hence it is not necessary
to transmit this value in all packets.

These packets cannot be used if other fields than the sequence
number, timestamp and identification changed. The timestamp also has
to change according to the following equations:

TS(n) = TS(n-1) + (SN(n) - SN(n-1)) * ddTS

   ddTS     : delta Timestamp
   TS(n)    : Timestamp of current packet
   TS(n-1)  : Timestamp of previous packet
   SN(n)    : Sequence Number of current packet
   SN(n-1)  : Sequence Number of previous packet

In this state the compressor might use the Two-Byte-Header or Three-
Byte-Header packet. These packets contain only the LSB of the RTP
Sequence Number, LSB of IPv4 identification and the keyword, which is
enough information for the decompressor to regenerate the original
header.

The packet format for the Two-Byte-Header packet is given below:

  0   1   2   3   4   5   6   7
+...+...+...+...+...+...+...+...+
```



```
:       MSB of Session CID      :  if 16 bit CID is used
+...+...+...+...+...+...+...+...+
:       LSB of Session CID      :  if 16 or 8 bit CID is used
+---+---+---+---+---+---+---+---+
| 0 |KW |LSB RTP Sequence Number|
+---+---+---+---+---+---+---+---+
|    LSB IPv4 Identification    |
+---+---+---+---+---+---+---+---+


The packet format for the Three-Byte-Header packet is given below:

  0   1   2   3   4   5   6   7
+...+...+...+...+...+...+...+...+
:       MSB of Session CID      :  if 16 bit CID is used
+...+...+...+...+...+...+...+...+
:       LSB of Session CID      :  if 16 or 8 bit CID is used
+---+---+---+---+---+---+---+---+
| 1 | 0 |KW | T |               |
+---+---+---+---+               +
|       LSB RTP Sequence Number |
+               &               +
|       LSB IPv4 Identification |
+---+---+---+---+---+---+---+---+

The T-bit indicates how the next bits are used to transport the RTP
Sequence Number and the IPv4 Identification:
   T=0:
     10 bit RTP Sequence Number
```

```
     10 bit IPv4 Identification

   T=1:
     8 bit RTP Sequence Number
     12 bit IPv4 Identification
```

The decision which of these packets is to be used should be done according to the number of packets already sent after the last update packet (or the first update packet of a set of update packets sent successively). The not transmitted MSB of these values must not have changed.

5.10.4.3 Context State packet

The use and format of the context state packet is similar to packet-profile 1, see section 5.10.3.3 for details.

6. Implementation issues

This document specifies mechanisms for the protocol, while much of the usage of these mechanisms is left to the implementors to decide upon. This chapter is aimed to give guidelines, ideas and suggestions for implementing the scheme.

6.1. Reverse decompression

This chapter describes an optional decompressor operation to reduce discarded packets due to an invalid context.

Once a context becomes invalid (e.g., in the case when more consecutive packet losses than expected has occurred), subsequent compressed packets cannot be decompressed correctly with normal decompression operation. This decompression operation aims at decompressing these packets with a later recovered context. The decompressor stores them until the context is validated. After the context is updated, the decompressor tries to recover the stored packets in the reverse order from the packet updating the context. Each time the stored packet is decompressed, its correctness is verified using the header compression CRC, which is transmitted in each compressed header. Correctly decompressed packets are transferred to upper layers in the original order.

Note that this reverse decompression introduces buffering while waiting for the context to be validated and thereby introduces additional delay. Thus, it should be used only when some amount of delay could be accepted. For example, for video packets belonging to the same video frame, the delay of packet arrival time does not cause presentation time delay. Delay-insensitive streaming applications can also be tolerant to such delay.

The following illustrates the decompression procedure in some detail:

1. The decompressor stores compressed packets that cannot be decompressed correctly due to an invalid context.

2. When the decompressor has received a context updating packet and the context has been validated, it starts to recover the stored packets in reverse order. Decompression is carried out followed by the last decompressed packet to its previous packet as if the two packets were reordered. After that, decompressor checks the correctness of the reconstructed header using the header compression CRC.

3. If the header compression CRC indicates a successful decompression, the decompressor stores the complete packet and tries to decompress its previous packet. In this way, the stored packets are recovered from correctly decompressed packets until no compressed packets are left. For each packet, the decompressor

checks the correctness of the decompressed headers using header compression CRC.

4. If the header compression CRC indicates an incorrectly decompressed packet, the reverse decompression attempt must be terminated and all remaining packets must be discarded.

5. Finally, the decompressor forwards all the correctly decompressed packets to upper layers in the original order.

6.2. Pre-verification of CRCs

For reasons of compression efficiency, it is desirable to keep the size of the header compression CRC as small as possible. However, if the size of the CRC is decreased, the reliability is also decreased and erroneous headers could be generated and passed on from the decompressor. It would then be desirable to find a method of increasing the strength of the CRC without making it larger.

There is one property of the header compression CRC and its usage that can be used to achieve this goal. The CRCs that will occur at the decompressor will in most cases follow a pattern well known also to the compressor. There are two factors that will affect which CRCs are generated and in which order they will occur. If the decompressor makes several reconstruction attempts, the first factor affecting the CRCs will be the order and properties of the assumptions made for each reconstruction attempt. The attempts are in general:

1:st attempt:	No loss is assumed
2:nd attempt:	Loss of the preceding packet is assumed
3:rd attempt:	Loss of the two preceding packets is assumed
4:th attempt:	Loss of the three preceding packets is assumed
etc.	

The other factor that will affect the CRCs generated is what has really happened to preceding packets, that is, if no loss has occurred or if one or several preceding packets have been lost between compressor and decompressor.

Since the compressor knows how the decompressor performs the reconstruction attempts, the compressor can PRE-CALCULATE and VERIFY the most probable CRC situations. If a CRC is found not to detect an erroneous header, then a different packet type with a larger CRC (such as the "normal" COMPRESSED packet) should be used instead or additional information could be sent (by using EXTENDED_COMPRESSED or DYNAMIC packets). To ensure reliability, the important thing is that the CRC must fail if the header is not correctly reconstructed. Combining the two factors described above gives a list of the most probable CRCs that MUST fail.

- If ONE packet WAS lost, attempt one (no loss) MUST fail
- If TWO packets WERE lost, attempt one (no loss) MUST fail
- If TWO packets WERE lost, attempt two (one lost) MUST fail
- If THREE packets WERE lost, attempt one (no loss) MUST fail
- If THREE packets WERE lost, attempt two (one lost) MUST fail
- If THREE packets WERE lost, attempt three (two lost) MUST fail
- etc.

By doing PRE-CALCULATIONS of the six CRCs that would be the result of the events listed above, the CRC can be kept strong enough, even with a reduced size, because CRCs likely to fail will be avoided.

6.3. New reconstruction attempts with LSB and LSP encoding

ROCCO profiles using LSP encoding can handle 25 consecutive packet losses without invalidating the context. LSB or LSP encoding is also used for other fields and the range handled is then much larger. However, for all LSP or LSB decoding, the range can be extended with multiples by making reconstruction attempts (also called "guesses"). The limiting factors are implementation complexity and time. The following example shows how this can be done:

In chapter 5.3.2, LSP encoding is described. When an LSP encoded value for M code-points is decoded to a value S'(n), the original header can be reconstructed. If the CRC verification fails, a new reconstruction attempt could be made with S'(n)+M as the sequence number. If M was a multiple of 2 (LSB encoding), this would be the same as changing the value of the lowest MSB bit (i.e. the lowest bit NOT transmitted in LSB). More attempts could then be made increasing the sequence number by M for each attempt.

7. Further work

(Editor: This section is _further work_ in particular as it needs to be integrated into the rest of the document.)

7.1. Compression of IPv6 extension headers

The ROHC scheme defined in this document currently do not support compression of IPv6 extension headers, which is an undesirable limitation. Therefore, it is necessary to investigate what is really needed from the compression scheme regarding compression of extensions, and also to further develop the scheme to include the desired extension support.

1. Header Compression for IPv6 Extension Header

The IPv6 extension headers are encoded as a list of items. Each item is one of the extension headers. The length of each extension header may vary from each other. When more than one extension header is used in the same packet, the order of these extension headers is recommended in RFC 2460, but not mandatory. Thus, although it is unlikely to happen, the order of the extension headers may vary during the same session. In addition, one or more extension header may be added or removed during the session and the content of each extension header may change. Therefore, the IPv6 extension headers are classified as a list of items and the item list compression mechanism can be applied.

The compression of IPv6 extension headers at the list level is specified in the item list compression scheme in Appendix COMPLIST. The compressed value of the extension header list is referred to as a compressed extension header list. The compression of IPv6 extension headers at the item level, i.e., the compression scheme used for each type of extension header, is defined in this subsection. The reference extension header used to compress a given extension header is the extension header in the reference list that has the same type. The compressed value of an extension header is referred to as a compressed extension header.

1.1. IPv6 Extension Header Types

The table below summarizes classification of the various IPv6 extension header fields. HbHH, DOH1, RH, FrH, AH, ESPH, DOH2, BU, BA, BR, and HA stand for Hop-by-Hop Options Header, Destination Options Header 1, Routing Header, Fragment Header, Authentication Header, Encapsulating Security Payload Header, Destination Options Header 2, Binding Update, Binding Acknowledgement, Binding Request and Home Address respectively.

Ext.	Static	Non-static

Header Type		Essential	Non-Essential
HbHH			Next Header Hdr Ext Len Options
DOH1			Next Header Hdr Ext Len Options
RH			Next Header Hdr Ext Len Routing Type Segments Left type-specific data
RH			Next Header Hdr Ext Len Routing Type Segments Left type-specific data
FrH	Reserved Res	identification Fragment Offset M flag	Next Header
AH		Sequence Number Authentication data	Next Header Payload Len SPI
ESPH*		Sequence Number	SPI
DOH2			Next Header Hdr Ext Len
BU	Option Type Reserved	Sequence Number	Option Length A, H, R, D Prefix Length Lifetime Sub-Options
BA	Option Type	Sequence Number	Option Length Status Lifetime Refresh Sub-Options
BR	Option Type		Option Length Sub-Options

```
  |   | HA | Option Type |                  | Option Length      |
  |   |    | Home Address|                  | Sub-Options        |
  +---+----+-----------------+--------------+--------------------+
```

* note: Only the fields that can be compressed are listed.

1.2. Compression of IPv6 Extension Headers at Item Level

For a given extension header in the extension header list, it can be classified as belonging to one of the three transformation cases defined in Appendix COMPLIST. Depending on the transformation case, the correspondent encoding technique is used. Note that the type-specific data field in the c_item with code "10" and "11" is not present.

1.2.1 Special Treatment of Next Header Field

The next header field in an extension header changes whenever the type of the immediately following header changes, e.g., a new extension header is inserted after it, the immediate subsequent extension header is removed from the list, or the order of several extension headers is changed. Thus, in particular, it may not be uncommon that for a given extension header, only the next header field changes but the remaining fields don't change. Therefore, the next header field in each extension header needs to be treated in a special way.

The classification of the transformation case that an extension header belongs to should depend on the behavior of the other remaining fields except the next header field. In the case that only the next header field changes, the extension header should be considered as unchanged, and classified as belonging to transformation case A. In the other case where both the next header field and some remaining fields change, the compression of the remaining fields for each type of the extension header is specified in section 1.2.2. The special treatment of the change of the next header field is defined as follows.

* In the case that a subsequent extension header is removed from the list or the order of several extension headers is changed, the new value of the next header field can be obtained from the reference extension header list. For example, assume that the reference extension header list (ref_list) consists of extension header A, B and C (ref_ext_hdr A, B, C), and the current extension header list (curr_list) only consists of extension headers A and C (curr_ext_hdr A, C). The order and value of the next header field of these extension headers are as follows.

```
ref_list:
+--------+------+   +--------+-----+   +--------+------+
| type B |      |   | type C |     |   | type D |      |
```

```
+----------+      |    +----------+      |    +----------+      |
|                 |    |                 |    |                 |
+-----------------+    +-----------------+    +-----------------+
  ref_ext_hdr A          ref_ext_hdr B          ref_ext_hdr C

curr_list:
+----------+------+    +----------+------+
| type C |        |    | type D          |
+----------+      |    +----------+      |
|                 |    |                 |
+-----------------+    +-----------------+
  curr_ext_hdr A         curr_ext_hdr C
```

Comparing the curr_ext_hdr A in curr_list and the ref_ext_hdr A in ref_list, the value of next header field is changed from "type B" to "type C" because of removal of extension header B. The new value of the next header field in curr_ext_hdr A, i.e., "type C" doesn't need to be sent to the decompressor, because when the decompressor detects (by observing the list level encoding) that the immediate following extension header B is removed from the list, it retrieves the next header field in ref_ext_hdr B and use it to replace the next header field in the curr_ext_hdr A.

A similar scheme is used to regenerate the next header field when the order of several extension headers is changed.

* In the case that a new extension header is inserted after an existing extension header, the next header field in the new extension header carries the type of itself, instead of the type of extension header that follows. For example, assume that the reference extension header list (ref_list) consists of extension header A and C (ref_ext_hdr A, C), and the current extension header list (curr_list) consists of extension header A, B and C (curr_ext_hdr A, B, C). The order and the value of the next header field of these extension headers are as follows.

```
ref_list:
+----------+------+    +----------+------+
| type C |        |    | type D |        |
+----------+      |    +----------+      |
|                 |    |                 |
+-----------------+    +-----------------+
  ref_ext_hdr A          ref_ext_hdr C

curr_list:
+----------+------+    +----------+------+    +----------+------+
| type B |        |    | type C |        |    | type D |        |
+----------+      |    +----------+      |    +----------+      |
|                 |    |                 |    |                 |
+-----------------+    +-----------------+    +-----------------+
  curr_ext_hdr A         curr_ext_hdr B         curr_ext_hdr C
```


Comparing the curr_list and the ref_list, the value of the next header field in extension header A is changed from "type C" to "type B".

In the compressed extension header list, the uncompressed curr_ext_hdr B is carried in the uncompressed data field in c_item or u_item depending on the list encoding scheme used. However, instead of carrying the type of the next header (type C) in the next header field, the type of curr_ext_hdr B (type B) should be carried. When the decompressor detects (by observing the list level encoding) that a new extension is inserted after curr_ext_hdr A, it will replace the old next header field in ref_ext_hdr A with the type of the inserted extension header, i.e., type B, which is carried in the next header field in the c_item or u_item for extension header B. At the same time, the decompressor also replace the next header field in curr_ext_hdr B with the old value of the next header field in ref_ext_hdr A, i.e., type C.

1.2.2. Compression of Each Type of Extension Header

In general, the encoding scheme used for each IPv6 extension header is similar to the scheme used for IPv4 and IPv6 base header, although the compressed format for each type of extension header may be different for each header. In this section, the format of the compressed data field in c_item or uc_item is defined for each type of extension header. Note that the non-essential fields discussed in the following subsections don't include the next header field.

1.2.2.1. Hop-by-Hop Options Header and Destination Options Header 1

The hop-by-hop options header (HbHH) and the destination option header 1 (DOH1, the header processed by the first destination that appear in the IPv6 Destination Address field plus subsequent destinations listed in the Routing header) are expected to rarely change from packet to packet during the session. However, if any change happens to any field in these two headers, the correspondent compressed extension header is sent.

The compressed HbHH consists of a bit mask that indicates the presence of the changed field, and the corresponding field value. The compressed DOH1 has the same structure as the compressed HbHH. The meaning of each field in the compressed HbHH is also the same as for the compressed DOH1. Therefore, in this subsection, only the compressed HbHH is discussed.

The structure of compressed HbHH is as follows.

```
                          +---+---
+:::::::::::::+::::::::::::::::::::::::::+
```



```
   compressed HbHH:   | L | O | Hdr Ext Len | Compressed Option List
  |
                          +---+---
+:::::::::::::+::::::::::::::::::::::::::+
```

* L bit indicates the presence of the Hdr Ext Len field that carries the value of Hdr Ext Len in the current HbHH.

* O bit indicates the presence of the Compressed Option List field that carries the compressed value of the Options field.

The Options field in HbHH is encoded as a list of options, and each option is considered as an item. The Options field can be compressed using the generic item list compression scheme specified in Appendix COMPLIST at the list level. At the item level, the format of the compressed option depends on the type of the option.

1.2.2.2. Routing Header

The content of the Routing Header (RH) is expected to rarely change from packet to packet during the session. However, if any change happens to any field in RH, a compressed RH is sent.

The structure of the compressed RH is as follows.

```
                  +---------------+-------------+--------------+
compressed RH:    | L | T | S | T'| Hdr Ext Len | Routing Type |
                  +---------------+-------------+--------------+

+---------------+---------------------------------------------------+
 Segments Left | type-specific data (compressed or uncompressed) |
+---------------+---------------------------------------------------+
```

The 4-bit bit mask indicates which fields are present.
* L bit - Hdr Ext Len
* T bit - Routing Type
* S bit - Segments Left
* T' bit - type-specific data

The Hdr Ext Len, Routing Type and Segments Left fields are all sent uncompressed. The type-specific data can be sent compressed or uncompressed. The type 0 routing header can be compressed using the scheme specified below and for any other unknown type of routing header, the type-specific data field should be sent uncompressed.

1.2.2.2.1. Compression of Type-specific Data Field in Type 0 Routing Header

The type-specific data field in the type 0 routing header consists of Reserved field and a list of 128-bit addresses. The Reserved field is not expected to change and doesn't need to be sent in the compressed type-specific data field. The list of addresses is encoded

as an item list and can be compressed using the scheme defined in Appendix COMPLIST. The structure of compressed type-specific data fields in the type 0 routing header is as follows.

```
compressed type-specific data field in type 0 routing header:
+---+------------------------------------------+
| C | compressed / uncompressed address list |
+---+------------------------------------------+
```

C bit indicates the type of the following field. "0" indicates that the uncompressed address list is carried in the following field, while "1" indicates the compressed address list is carried. The decision of which format to use is up to the compressor. An example of the criteria could be compression efficiency or processing complexity.

As mentioned before, the address list can be compressed using the scheme defined in Appendix COMPLIST. Each address in the address list is considered as an item. The insertion and removal scheme defined in Appendix COMPLIST can be used to encode the change.

1.2.2.3. Fragment Header

If the fragment header (FrH) is present, its contents are expected to change from packet to packet. To address the change, a compressed FrH is sent.

The structure of the compressed FrH is as follows.

```
                   +-----------------+--------+---------------------------
------+
   Compressed FrH: | Fragment Offset | M flag | compressed
Identification |
                   +-----------------+--------+---------------------------
------
```

The Fragment Offset and M flag fields in the compressed FrH are copies of the same fields in the original FrH. The compressed Identification field carries the compressed value of the Identification field in the original FrH, using the Identification field in the reference FrH as the reference. The scheme used to compress and encode Identification field is VLE as defined in [ACT]. The format of compressed Identification using VLE is listed as follows.

- "0" + 4-bit LSB
- "10" + 8-bit LSB
- "110" + 16-bit LSB
- "111" + 32-bit LSB

1.2.2.4. Authentication Header and Encapsulating Security Payload Header

In the Authentication Header (AH), the SPI field only changes when a new security session is established, thus, it is expected to rarely change from packet to packet during the session. The Payload Len field changes only when SPI changes. Two change cases are listed as follow.

* In the case that the SPI field changes, all the fields in AH may change. For simplicity, an uncompressed AH is sent.

* In the case that no change happens to the SPI field, the AH is not considered as changed. When the decompressor detects from the encoding that the AH is not changed, it copies the SPI and Payload Len fields from the reference AH. The other two fields in AH - sequence number and authentication data, are handled as defined below.

The sequence number field in the AH contains a monotonically increasing counter value for a security association. Like IP-ID in Ipv4, if one observes only one of the flows, the sequence number in

AH may appear to be nonlinear due to disruption by other IP flows that also use the same security association. If the sequence number in the AH linearly increases, it doesn't need to be sent. The decompressor regenerates this field by applying linear extrapolation (with delta = 1). Otherwise, a compressed sequence number should be sent in a compressed IP/UDP/RTP header. The format of the compressed IP/UDP/RTP header containing the compressed sequence number should be defined in ROHC. The compression scheme for the sequence number in the AH is VLE, as defined in [ACE].

The authentication data field in AH changes from packet to packet and should be sent in every packet. If the uncompressed AH is sent, the authentication data field is sent inside the uncompressed AH; otherwise, it is sent after the compressed IP/UDP/RTP and IPv6 extension headers and before the payload.

If Encapsulating Security Payload Header (ESP) is used, the UDP and RTP headers are both encrypted and cannot be compressed. In this case, special compressed packet format needs to be defined in ROHC. In ESP, the only fields that can be compressed are the SPI and the sequence number. If SPI field changes, the uncompressed ESP is sent; otherwise, a compressed ESP that carries all the fields except SPI and sequence number is sent. The sequence number in ESP has the same behavior as the same field in AH, thus, they are compressed in the same manner.

1.2.2.5. Destination Options Header 2

The Destination Option Header 2 (DOH2) is for options to be processed only by the final destination of the packet. When ESP is used to provide security services, the DOH2 is encrypted and cannot

be compressed. Otherwise, the following compression mechanisms can be applied.

DOH2 contains Hdr Ext Len and Options fields. If any change happens to any fields in DOH2, a compressed DOH2 is sent.

The structure of the compressed DOH2 is as follows.

```
                  +-------------+-------------------------------+
compressed DOH2:  | Hdr Ext Len | compressed options list     |
                  +-------------+-------------------------------+
```

The Hdr Ext Len in the compressed DOH2 is a copy of the same field in the original DOH2. The compressed options list field carries the compressed value of the options field. Multiple options can be included in the options field. Assuming that the number of options in the options field in DOH2 is n, the structure of compressed options list field is as follows.

```
                      +------------+------------+-----+------------+
compressed options:   | c_option 1 | c_option 2 | ... | c_option n |
                      +------------+------------+-----+------------+
```

The i-th compressed option (c_option i) in the compressed options list carries the compressed or uncompressed value of the i-th option in the options field in DOH2. The structure of c_option is defined as

follows.

```
             +--------------+---+----------------------------------------+
   c_option: | Option Type  | C | compressed / uncompressed option       |
             +--------------+---+----------------------------------------+
```

* Option Type is the copy of the same field in the uncompressed option. Four destination options - Binding Update (BU), Binding Acknowledgement (BA), Binding Request (BR) and Home Address (HA) have been defined in mobile IPv6.

* C bit = "0" indicates the all the fields in the option except the Option Type are sent, while C bit = "1" indicates the compressed option as followed. The decision of sending compressed option or uncompressed option as well as the format of the compressed option for BU, BA, BR and HA are defined in the following subsections. For any other unknown type of options, the uncompressed value is always sent.

Since each of the aforementioned four options follows a certain pattern individually, but is not sent in every packet, an individual context for each type of option should be established and maintained by the compressor and the decompressor. The linkage between these individual contexts and the context maintained for IP base header and the UDP/RTP header could be the RTP sequence number. In addition, an individual compression state is maintained for each option. Unlike

the compression states used for IP base header and RTP/UDP headers, only two compression states are defined for these four options - original state and compressed state. In the original state, the original value of the option is sent, while in the compressed state, the compressed value of the option is sent.

3.2.6.5.1. Home Address Option (HA) and Binding Request Option (BR)

At the beginning, the compressor stays in the original state and sends uncompressed HAs. When the decompressor receives an uncompressed HA, it restores the static fields, i.e., the option Type field and the Home Address field, and then acknowledges the received packet. After receiving an acknowledgement for the packet that carries the HA, the compressor switches to the compressed state for HA. When the decompressor receives a compressed HA, it retrieves the static fields from storage. The sub-option field (if present) and the Option Len field can be obtained from the compressed HA.

The structure of compressed HA is defined as follows.

```
                   +---+------------+------------+
   compressed HA:  | S | Option Len | sub-option |
                   +---+------------+------------+
```

S bit indicates whether or not sub-option is present. If it is present, both the Option Len and sub-option fields should be sent uncompressed and the S bit is set to "1". Otherwise, S bit is set to "0" and no other fields needs to be sent in the compressed HA. In the second case, the decompressor regenerates the Option Len field as 16.

BR option can also be sent compressed or uncompressed. The compression scheme for BR option is similar to that of HA option. The structure of compressed BR is the same as the structure of compressed HA. The only difference is in the case that the sub-option field is not present, the decompressor assigns 0 to the Option Len field.

1.2.2.5.2. Binding Update Option (BU) and Binding Acknowledgement Option (BA)

BU and BA options are not sent in every packet. For example, BU option is only sent when the mobile node changes its care-of address or it observes that its entry in the binding cache at the correspondent node doesn't exist or may expire soon. Since these options are not sent frequently, to keep a simple compression and decompression logic, these options can be sent uncompressed whenever they are present.

However, to obtain higher compression efficiency, these options can be sent compressed at the price of more complicated logic and a higher memory requirement.

To compress the current BU option, a BU that was sent previously and has been acknowledged by the decompressor is used as the reference. Since the BU option is not sent in every packet, the reference header used to compress the base IPv6 header and the UDP/RTP header may not be able to be used as the reference for BU option because it may not contain a BU option. Therefore, a separate reference needs to be maintained for BU option.

The reference for BU can be selected based on the acknowledgement. When the decompressor receives a packet containing BU, it inserts the BU into the sliding window (refer to [ACE]) in the individual BU context, and acknowledges the packet. After the compressor receives the acknowledgement, it updates the reference to be used for BU.

When the BU is present in the packet the next time, the new reference is used to compress the BU. To identify the reference BU, an identifier for BU (BU_ref_id) is needed. The sequence number field in BU option increments (not necessary strictly by 1) from packet to packet and can be used as the BU_ref_id. The BU_ref_id is carried in the compressed BU header.

When the decompressor receives a compressed BU header, it retrieves the reference BU from the sliding window and use it to decompress the BU option. The decompressor also shrinks the sliding window by removing all the BUs before the reference BU.

The structure of the compressed BU is as follows.

```
                    +-----------+---------------+-----------+--------
-+
   compressed BU: | BU_ref_id | A | R | K | D | L | PL | LT | SP | SC
                    +-----------+---------------+-----------+--------
-+
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-------------------------------+
 Option Length | Prefix Length | Compressed Sequence Number |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-------------------------------

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 Compressed Lifetime | Sub-Options |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The A, H, R, D bits are copied from the original BU. The subsequent bit mask indicates the presence of the field that is changed. The mapping of the bit mask and the subsequent changing fields is as follows. A "1" in the bit mask means the correspondent field is present.

- L bit: Option Length

- PL bit: Prefix Length
- LT bit: Lifetime

Among the above three fields, only the Lifetime field can be sent compressed if it changes. All the other fields should be sent uncompressed. The compression scheme used for Lifetime field is VLE as defined in [ACE]. The format of compressed Lifetime is the same as the format for compressed Identification in Fragment Header, which is defined in 1.2.3.3.

The SP and SC bits are used for compression of the sub-options field. SP bit indicates the presence of sub-options field. If it is present, the SC bit indicates whether or not the sub-options field can be sent compressed. If the sub-options field in the current BU is not the same as the sub-options field in the reference BU, SC bit is set to "0" and the uncompressed value of the sub-options field is sent. Otherwise, SC bit is set to "1" and sub-options field is not sent.

The sequence number in the BU should be sent all the time. Its compressed value is carried in the compressed sequence number field. The compression scheme for sequence number is VLE, as defined in [ACE]. The format of the compressed sequence number is as follow.

- "0" + 4-bit LSB
- "10" + 8-bit LSB
- "11" + 16-bit LSB

The compression scheme for BA option is exactly the same as the scheme defined for BU option. The format for the compressed BA is as follows.

```
               +------------+-----------------------+----------+
compressed BA: | BA_ref_id | L | ST | LT | R | SP | SC |
               +------------+-----------------------+----------+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-------------------------------+
 Option Length | Status | Compressed Sequence Number |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-------------------------------+
```

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 Compressed Lifetime | Compressed Refresh | Sub-Options
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

BU_ref_id is used to identify the reference BA. The sequence number field in BA option increments (not necessary strictly by 1) from packet to packet and can be used as the BA_ref_id.

The mapping of the bit mask and the subsequent changing fields is as follows.

- L bit: Option Length
- S bit: Status
- LT bit: Lifetime
- R bit: Refresh

Among the above four fields, Lifetime and Refresh fields can be sent compressed if they change. The other two fields should be sent uncompressed. The compression scheme used for Lifetime field or Refresh field is VLE as defined in [ACE]. The format for compressed Lifetime and compressed Refresh is the same as the format for compressed Identification in Fragment Header, which is defined in 7.2.3.3.

The sequence number in BA has the same behavior as the sequence number in the BU, thus, it is compressed in the same manner.

7.2. Efficient compression of CSRC lists

The Contributing Source (CSRC) List in a RTP header contains the Synchronization Source (SSRC) identifiers of the contributing sources for the payload in current packet.

A CSRC list contains at most 15 identifiers, due to the 4-bit size of CSRC Count (CC) field in RTP header. Each 32-bit identifier is chosen randomly by the original synchronization source so that it is globally unique within an RTP session.

The compression scheme introduced here will utilize the facts mentioned above. To maintain transparency, the order of identifiers is preserved during compression. In other words, the CSRC list is really compressed as a list, not as a set.

7.2.1. Data Structure and Algorithm

The scheme is essentially reference based compression. (Refer to Appendix COMPLIST for general discussion on list compression). The reference CSRC list (ref_CSRC) could be the CSRC list in the last acknowledged RTP header.

Four encoding formats are provided in this scheme, which are differentiated by the Encoding Type (ET) value. The compressor will choose the most efficient one based on the change from the ref_CSRC to the current CSRC list (cur_CSRC).

7.2.2. Generic

This format can handle any change from the ref_CSRC to the cur_CSRC. However, it should be used only if the change cannot be handled efficiently by the formats described in the following sections.

Below is the format of a compressed CSRC list (comp_CSRC). Note that the ref_CSRC is identified by the RTP SN of the RTP header in which it was carried.

```
+---------+--------+--------+-----------+------+------------+
| ET = 00 | ref_SN | cur_CC | c_item 1  | .... | c_item n   |
+---------+--------+--------+-----------+------+------------+
```

```
ET:      2 bits
ref_SN:  could be uncompressed (16 bits), or compressed (a few LSBs),
         depending on the ROHC compression of RTP SN
cur_CC:  4 bits, the number of c items
```

Each c_item above has one of the following structures:

a)

```
+---+-----+
| 0 | pos |
+---+-----+
```

This indicates that an SSRC at position pos in the ref_CSRC is also present in cur_CSRC. Note the length of the pos field needs not to be sent explicitly, since it can be determined by both compressor and decompressor as ceil(log2(k)), where k is the number of SSRCs in the ref_CSRC.

b)

```
+---+------+
| 1 | SSRC |
+---+------+
```

This indicates a new SSRC is present in current CSRC list. Note the new SSRC itself is not compressed due to its random nature.

After receiving a comp_CSRC, the decompressor 1) fetches the ref_CSRC from its context, 2) scans the c_item list in the received comp_CSRC and builds the cur_CSRC item by item, and 3) copies the value of cur_CC into the CC field in the decompressed RTP header.

7.2.3. Insertion Only

If the change from the ref_CSRC to cur_CSRC only involves addition of new SSRCs (i.e., no order change, no deletion), a more efficient

format can be used.

For example, this format is efficient in handling the event that a new party or parties join speaker becomes active in a conference call.

```
+--------+---------+--------+-------+-------+-------+
| ET = 01 | ref_SN | add_CC | pos 1 | ... | pos n |
+--------+---------+--------+-------+-------+-------+

+--------+-----+--------+
 SSRC 1 | ... | SSRC n |
+--------+-----+--------+
```

ET: 2 bits
ref_SN: same as in generic format
add_CC: 4 bits, the number of new SSRCs present in this comp_CSRC.

After receiving a comp_CSRC with ET = 01, the decompressor will insert the new SSRC_i right before position pos_i in the ref_CSRC.

Note that the length of pos fields is now equal to ceil(log2(k+1)), where k is the number of SSRCs in the ref_CSRC. The extra one is needed since the insertion could happen at both the head and the tail of the list.

7.2.4. Deletion Only

This format is optimized for the case that the change from ref_CSRC to the cur_CSRC only involves removal of some SSRC(s) in the ref_CSRC. For example, it can be used when a party or parties leave a conference call.

```
+---------+---------+--------+-------+-------+-------+
| ET = 10 | ref_SN | del_CC | pos 1 | ... | pos m |
+---------+---------+--------+-------+-------+-------+
```

ET: 2 bits
ref_SN: same as in generic format
del_CC: the number of SSRCs should be deleted from ref_CSRC.
          length = log2(k), where k is the number of SSRCs in
          the ref_CSRC.

After receiving such a comp_CSRC, the decompressor will fetch the ref_CSRC, then remove each SSRC whose position is listed in the comp_CSRC.

The length of pos fields are determined in the same way as the generic format.

7.2.5. Insertion and Deletion Only

The following format can handle the case where both insertion and deletion of SSRCs happen, but there is no order change. Note that the generic format could be more efficient if the change is significant compared with the size of cur_CSRC, and thus should be used instead.

```
+---------+-----------------+--------------+-------+
| ET = 11 | ref_SN | del_CC | pos 1 | ... | pos m |
+---------+--------+--------+-------+-----+-------+

-----+------------+-------------+--------+--------+
 add_CC | pos 1 | ... | pos n | SSRC 1 | ... | SSRC n |
-------+-------+-----+-------+--------+-----+--------+
```

ET: 2 bits
ref_SN: same as in generic format
del_CC: same as in the deletion only fomrat
add_CC: 4 bits, the number of new SSRCs

This case can be considered as two combined transformations. First, deletion (section X.2.3) is applied to ref_CSRC as identified by the ref_SN. Let's call the resulted CSRC list mid_CSRC. Then, insertion (section X.2.2) is applied to mid_CSRC. Therefore, each pos in the deletion part refers a position in ref_CSRC, while each pos in the insertion part indexes into the mid_CSRC.

7.3. Tunneling

7.3.1. Header Compression for IPv4 Tunneling Header

In order to route the packets to the mobile node that is on a foreign link, the home agent of the mobile node may encapsulate the original packet into an IP header and tunnel the packet to the care-of address of the mobile node. In the case of foreign agent care-of address in Mobile IPv4, the tunneling header in each tunneled packet will be removed by the foreign agent before transferring it to the mobile node through the air interface; therefore there is no need for compression of tunneling header. In the case that mobile node uses collocated care-of address, the tunneled packet will be sent to mobile station through air interface, and compression needs to be applied to the tunneling header.

7.3.1.1. Mobile IPv4 Tunneling Header Fields Type

The table below summarizes classification of the various fields defined in different tunneling headers used in Mobile IPv4. In the column of Encapsulation Scheme (Enc. Scheme), three encapsulation

methods are included - IP in IP Encapsulation (IIE), Minimum

Encapsulation (ME), Generic Routing Encapsulation (GRE).

(Editor's note: Harmonize with the way this is described in ROHC document)

Enc. Scheme	Header type	Static	Non-static: Essential	Non-static: Non-Essential
IIE	inner header	same as in the table in Appendix A in ACE Internet Draft		
	outer header	same as in the table in Appendix A in ACE Internet Draft		
ME	IP header	same as in the table in Appendix A in ACE Internet Draft		
	Mini. Fw. header	Protocol		S bit Header Checksum Original Dest. Addr. Original Src. Addr.
GRE	inner header	same as in the table in Appendix A in ACE Internet Draft		
	outer header	same as in the table in Appendix A in ACE Internet Draft		
	GRE header	Protocol Ver	Sequence number	C, R, X, S, s bits Recur Flags Checksum Offset Key Routing

7.3.1.2. Compression of Tunneling Headers in MIPv4

Three encapsulation schemes have been specified in MIPv4. For different encapsulation scheme, the compression methods are different from each other.

7.3.1.2.1. IP in IP Encapsulation in IPv4

Using IP in IP Encapsulation, the original inner IP header is not modified at all and therefore can be compressed as if it is not

encapsulated. The outer header is compressed at the IP level, while the inner header is compressed as defined in ROHC.

7.3.1.2.2. Minimum Encapsulation in IPv4

With Minimum Encapsulation, the original IP header is modified and

the Minimal Forwarding Header is inserted between the modified IP header and the original IP payload. The modified IP header plus the the UDP/RTP headers is compressed as defined in ROHC.

The compression scheme for the Minimal Forwarding Header is similar to the scheme applied to the IP header. The static and changing non- essential fields in the Minimum Forwarding Header are sent in the Full Header and Refresh state. When any change happens to any non-essential field in the Minimum Forwarding Header, a compressed header with a bit mask indicating the change should be sent.

7.3.1.2.3. Generic Routing Encapsulation in IPv4

With Generic Routing Encapsulation, the original IP packet is encapsulated in an outer IP header. A GRE header is inserted between the inner header and the outer header. The original IP/UDP/RTP header is compressed as if there is no encapsulation. The outer IP header is compressed at the IP level.

The compression scheme for the GRE header is similar to the scheme applied to the IP header. All the static and changing non essential fields in the GRE header are sent in the Full Header and refresh state. When any change happens to any non-essential field in the GRE header, a compressed header with a bit mask indicating the change should be sent. If the sequence number in the GRE header is present, the scheme to compress sequence number could be VLE, as defined in [illegible] draft.

7.4. RTCP

RTCP is the RTP Control Protocol, [RTP]. RTCP is based on periodic transmission of control packets to all participants in a session, using the same distribution mechanism as for data packets. Its primary function is to provide feedback from the data receivers on the quality of the data distribution. The feedback information may be used for issues related to congestion control functions, and directly useful for control of adaptive encodings.

In an RTP session there will be two types of packet streams; one with the RTP-header and application data, and a second stream with the RTCP control information. The difference between the streams at the transport level is the UDP port numbers, which is plus one for RTCP.

The question is now should ROHC header compressor handle the RTCP stream.

a) One compressor/decompressor entity for both streams and carried on the same channel using CIDs to distinguish between them. Although they could share some parts of their contexts. Hence, on the RTCP stream IP/UDP compression might be utilized.

b) One compressor/decompressor entity for both streams and carried on the same channel, but without using CIDs to distinguish between them. To avoid unnecessary extra overhead a packet type, or some other method, can be used to tell that this compressed packet carries RTCP

data and not RTP.
c)   Two compressor/decompressor entities, one for RTP and another one for RTCP, and the streams carried on their own channel. This means that they will not share the same CID number space.

7.5. non-RTP UDP traffic

(Editor's note: This is text from draft-koren-avt-crtp-enhance-01.txt to be added to rohc _ not yet consistent with the rest of the document)

2.1 The negative cache stream flag

Certain streams, known or suspected to not be RTP, can be placed in a "negative cache" at the compressor, so only the IP and UDP headers are compressed. It is beneficial to notify the decompressor that the compressed stream is in the negative cache: for such streams the context is shorter - there is no need to include the RTP header, and all RTP-related calculations can be avoided.

In this enhancement, a new flag bit "N" is added to the FULL_HEADER packet that initializes a context at the decompressor. The bit occupied by the new flag was previously always set to zero. If the N flag is set to 1, this indicates that no COMPRESSED_RTP packets will be transmitted in this context. This flag is only an optimization and the decompressor may choose to ignore it.

Format of the FULL_HEADER length fields with the negative cache flag:

For 8-bit context ID:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0|1| Generation|       CID     |  First length field
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|          0          |N|  seq  |  Second length field
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  N=1: negative cache stream
```


For 16-bit context ID:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|1|1| Generation|  0 |N|  seq  |  First length field
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+  N=1: negative cache stream

+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|              CID              |  Second length field
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

2.2 Reject a new compressed stream

In a point to point link the two nodes can agree on the number of compressed sessions they are prepared to support for this link. In an end-to-end scheme a host may have compressed sessions with many hosts and eventually may run out of resources. When the end-to-end tunnel is negotiated, the number of contexts needed may not be predictable. This enhancement allows the negotiated number of contexts to be larger than could be accommodated if many tunnels are established. Then, as context resources are consumed, an attempt to set up a new context may be rejected.
The compressor initiates a compression of a stream by sending a FULL_HEADER packet. Currently if the decompressor has insufficient resources to decompress the new stream, it can send a CONTEXT_STATE packet to invalidate the newly compressed stream. The compressor does not know the reason for the invalidation; usually this happens when the decompressor gets out of synchronization due to packet loss. The compressor will most likely reattempt to compress this stream by sending another FULL_HEADER.
This enhancement specifies that the decompressor may reject the compression of a stream by sending a REJECT message to the compressor. A REJECT message tells the compressor to stop compressing this stream.
The REJECT message is a CONTEXT_STATE message with an additional flag:

```
    Type code = 1 :CONTEXT_STATE for 8-bit CID streams
    Type code = 2 :  CONTEXT_STATE for16-bit CID streams
```

Here is the format of CONTEXT_STATE packets with REJECT flags:

```
                  0   1   2   3   4   5   6   7
                +---+---+---+---+---+---+---+---+
                |Type code=1: CS, 8-bit CID |
                +---+---+---+---+---+---+---+---+
                |         context count         |
                +---+---+---+---+---+---+---+---+
```



```
                |       session context ID      |
                +---+---+---+---+---+---+---+---+
                | 1 |R=1| 0 | 0 |   sequence    |   R is the REJECT flag
                +---+---+---+---+---+---+---+---+
                | 0 | 0 |      generation       |
                +---+---+---+---+---+---+---+---+
                               . . .
                +---+---+---+---+---+---+---+---+
                |       session context ID      |
                +---+---+---+---+---+---+---+---+
                | 1 |R=1| 0 | 0 |   sequence    |   R is the REJECT flag
                +---+---+---+---+---+---+---+---+
                | 0 | 0 |      generation       |
                +---+---+---+---+---+---+---+---+


                  0   1   2   3   4   5   6   7
                +---+---+---+---+---+---+---+---+
                |Type code=2: CS, 16-bit CID|
```

```
+---+---+---+---+---+---+---+---+
|         context count         |
+---+---+---+---+---+---+---+---+
:                               |
+       session context ID      +
|                               |
+---+---+---+---+---+---+---+---+
| 1 |R=1| 0 | 0 |   sequence    |   R is the REJECT flag
+---+---+---+---+---+---+---+---+
| 0 | 0 |      generation       |
+---+---+---+---+---+---+---+---+
              . . .
+---+---+---+---+---+---+---+---+
|                               |
+       session context ID      +
|                               |
+---+---+---+---+---+---+---+---+
| 1 |R=1| 0 | 0 |   sequence    |   R is the REJECT flag
+---+---+---+---+---+---+---+---+
| 0 | 0 |      generation       |
+---+---+---+---+---+---+---+---+
```

The session CID, sequence and generation are taken from the FULL_HEADER.

The compressor may decide to wait for a while before attempting to compress additional streams destined to the rejecting host.

8. Implementation status

(Editor: I don't think we need this section. We might want an _implementation notes_ section at one point.)

9. Security considerations

Because encryption eliminates the redundancy that header compression schemes try to exploit, there is some inducement to forego encryption in order to achieve operation over low-bandwidth links. However, for those cases where encryption of data (and not headers) is sufficient, RTP does specify an alternative encryption method in which only the RTP payload is encrypted and the headers are left in the clear. That would still allow header compression to be applied.

A malfunctioning or malicious header compressor could cause the header decompressor to reconstitute packets that do not match the original packets but still have valid IP, UDP and RTP headers and possibly even valid UDP checksums. Such corruption may be detected with end-to-end authentication and integrity mechanisms which will not be affected by the compression. Further, this header compression scheme provides an internal checksum for verification of re-constructed headers. This reduces the probability of producing decompressed headers not matching the original ones without this being noticed.

Denial-of-service attacks are possible if an intruder can introduce (for example) bogus STATIC, DYNAMIC or FEEDBACK packets onto the link and thereby cause compression efficiency to be reduced. However, an intruder having the ability to inject arbitrary packets at the link layer in this manner raises additional security issues that dwarf those related to the use of header compression.

TBW: Header compression and IPsec

10. Acknowledgements

When designing this protocol, earlier header compression ideas described in [CJHC], [IPHC] and [CRTP] have been important sources of knowledge.

Thanks to Takeshi Yoshimura at NTT DoCoMo for providing the reverse decompression chapter (chapter 6.3). Thanks also to Anton Martensson for many valuable draft contributions and to Andreas Jonsson (Lulea University), who made a great job supporting this work in his study of header field change patterns. Thanks also to all others who have given comments.

11. Intellectual property considerations

(Editor's note: this section will go to www.ietf.org/ipr and be replaced by the standard reference to that, but for now it is left in the draft to simplify working on it.)

This proposal is in conformity with RFC 2026.

Telefonaktiebolaget LM Ericsson and its subsidiaries, in accordance with corporate policy, will for submissions rightfully made by its employees which are adopted or recommended as a standard by the IETF offer patent licensing as follows:

If part(s) of a submission by Ericsson employees is (are) included in a standard and Ericsson has patents and/or patent application(s) that are essential to implementation of such included part(s) in said standard, Ericsson is prepared to grant - on the basis of reciprocity (grant-back) - a license on such included part(s) on reasonable, non-discriminatory terms and conditions.

For the avoidance of doubt this general patent licensing undertaking applies to this proposal.

Nokia has filed patent applications that might possibly have technical relation to this contribution.

Matsushita has filed patent applications that might possibly have technical relation to this contribution.
If part(s) of the contribution by Matsushita employee is (are)

included in a standard and Matsushita has patents and/or patent application(s) that are essential to implementation of such included part(s) in said standard, Matsushita is prepared to grant - on the basis of reciprocity (grantback) - a license on such included part(s) on reasonable, non-discriminatory terms and conditions (in according with paragraph 10.3.3 of the RFC 2026).

12. References

[UDP] Jon Postel, "User Datagram Protocol", RFC 768, August 1980.

[IPv4] Jon Postel, "Internet Protocol", RFC 791, September 1981.

[IPv6] Steven Deering, Robert Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RTP] Henning Schulzrinne, Stephen Casner, Ron Frederick, Van Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

[HDLC] William Simpson, "PPP in HDLC-like framing", RFC 1662, 1994.

[VJHC] Van Jacobson, "Compressing TCP/IP Headers for Low-Speed Serial Links", RFC 1144, February 1990.

[IPHC] Mikael Degermark, Bjorn Nordgren, Stephen Pink, "IP Header Compression", RFC 2507, February 1999.

[CRTP] Steven Casner, Van Jacobson, "Compressing IP/UDP/RTP Headers for Low-Speed Serial Links", RFC 2508, February 1999.

[PPPHC] Mathias Engan, Steven Casner, Carsten Bormann, "IP Header Compression over PPP", RFC 2509, February 1999.

[CRTPC] Mikael Degermark, Hans Hannu, Lars-Erik Jonsson, Krister Svanbro, "CRTP over cellular radio links", Internet Draft (work in progress), December 1999. <draft-degermark-crtp-cellular-01.txt>

[REQ] Mikael Degermark, "Requirements for robust IP/UDP/RTP header compression", Internet Draft (work in progress), June 2000. <draft-ietf-rohc-rtp-requirements-01.txt>

[LLG] Krister Svanbro, "Lower Layer Guidelines for Robust Header Compression", Internet Draft (work in progress), May 2000. <draft-ietf-rohc-lower-layer-guidelines-00.txt>

[CELL] Lars Westberg, Morgan Lindqvist, "Realtime traffic over cellular access networks", Internet Draft (work in progress), May 2000. <draft-westberg-realtime-cellular-02.txt>

[WCDMA] "Universal Mobile Telecommunications System (UMTS); Selection procedures for the choice of radio transmission technologies of the UMTS (UMTS 30.03 version 3.1.0)", ETSI TR 101 112 V3.0.1, November 1997.

13. Authors' addresses

Carsten Bormann
Tel: +
Fax: +
EMail:

Lars-Erik Jonsson
Ericsson Erisoft AB
SE-971 28 Lulea, Sweden
Tel: +46 920 20 21 07
Fax: +46 920 20 20 99
EMail: lars-erik.jonsson@ericsson.com

other authors....

Appendix A. Detailed classification of header fields

Header compression is possible due to the fact that most header fields do not vary randomly from packet to packet. Many of the fields exhibit static behavior or changes in a more or less predictable way. When designing a header compression scheme, it is of fundamental importance to understand the behavior of the fields in detail.

In this appendix, all IP, UDP and RTP header fields are classified and analyzed in two steps. First, we have a general classification in A.1 where the fields are classified based on stable knowledge and assumptions. The general classification does not take into account the change characteristics of changing fields because those will vary more or less depending on the implementation and on the application used. A less stable but more detailed analysis considering the change characteristics is then done in A.2. Finally, A.3 summarizes this appendix with conclusions about how the various header fields should be handled by the header compression scheme to optimize compression and functionality.

A.1. General classification

On a general level, the header fields are separated into 5 classes:

INFERRED These fields contain values that can be inferred from other values, for example the size of the frame carrying the packet, and thus does not have to be handled at all by the compression scheme.

STATIC These fields are expected to be constant throughout the lifetime of the packet stream. Static information must in some way be communicated once.

STATIC-DEF STATIC fields whose values define a packet stream. They are in general handled as STATIC.

STATIC-KNOWN These STATIC fields are expected to have well-known values and therefore do not need to be communicated at all.

CHANGING These fields are expected to vary in some way, either randomly, within a limited value set or range, or in some other manner.

In this section, each of the IP, UDP and RTP header fields is

assigned to one of these classes. For all fields except those classified as CHANGING, the motives for the classification are also stated. CHANGING fields are in A.2 further examined and classified based on their expected change behavior.

A.1.1. IPv6 header fields

Field	Size (bits)	Class
Version	4	STATIC-KNOWN
Traffic Class	8	CHANGING
Flow Label	20	STATIC-DEF
Payload Length	16	INFERRED
Next Header	8	STATIC-KNOWN
Hop Limit	8	CHANGING
Source Address	128	STATIC-DEF
Destination Address	128	STATIC-DEF

Version

The version field states which IP version the packet is based on. Packets with different values in this field must be handled by different IP stacks. For header compression, different compression profiles must also be used. When compressor and decompressor have negotiated which profile to use, the IP version is also known to both parties. The field is therefore classified as STATIC-KNOWN.

Flow Label

This field may be used to identify packets belonging to a specific packet stream. If not used, the value should be set to zero. Otherwise, all packets belonging to the same stream must have the same value in this field, it being one of the fields defining the stream. The field is therefore classified as STATIC-DEF.

Payload Length

Information about the packet length (and then also payload length) is expected to be provided by the link layer. The field is therefore classified as INFERRED.

Next Header

This field is expected to have the same value in all packets of a packet stream. As for the version number, a certain compression profile can only handle a specific next header which means that this value is known when profile has been negotiated. The field is therefore classified as STATIC-KNOWN.

Source and Destination addresses

These fields are part of the definition of a stream and must thus be constant for all packets in the stream. The fields are therefore classified as STATIC-DEF.

Summarizing the bits corresponding to the classes gives:

Class	Size (octets)
INFERRED	2
STATIC-DEF	34.5
STATIC-KNOWN	1.5
CHANGING	2

A.1.2. IPv4 header fields

Field	Size (bits)	Class
Version	4	STATIC-KNOWN
Header Length	4	STATIC-KNOWN
Type Of Service	8	CHANGING
Packet Length	16	INFERRED
Identification	16	CHANGING
Reserved flag	1	STATIC-KNOWN
May Fragment flag	1	STATIC
Last Fragment flag	1	STATIC-KNOWN
Fragment Offset	13	STATIC-KNOWN
Time To Live	8	CHANGING
Protocol	8	STATIC-KNOWN
Header Checksum	16	INFERRED
Source Address	32	STATIC-DEF
Destination Address	32	STATIC-DEF

Version

The version field states which IP version the packet is based on and packets with different values in this field must be handled by different IP stacks. For header compression, different compression profiles must also be used. When compressor and decompressor has negotiated which profile to use, the IP version is also well known to both parties. The field is therefore classified as STATIC-KNOWN.

Header Length

As long as there are no options present in the IP header, the header length is constant and well known. If there are options, the fields would be STATIC, but we assume no options. The field is therefore classified as STATIC-KNOWN.

Packet Length

Information about the packet length is expected to be provided by the link layer. The field is therefore classified as INFERRED.

Flags

The Reserved flag must be set to zero and is therefore classified as STATIC-KNOWN. The May Fragment flag will be constant for all packets in a stream and is therefore classified as STATIC. Finally,

the Last Fragment bit is expected to be zero because fragmentation is NOT expected, due to the small packet size expected. The Last

Fragment bit is therefore classified as STATIC-KNOWN.

Fragment Offset

With the assumption that no fragmentation occurs, the fragment offset is always zero. The field is therefore classified as STATIC-KNOWN.

Protocol

This field is expected to have the same value in all packets of a packet stream. As for the version number, a certain compression profile can only handle a specific next header which means that this value is well known when profile has been negotiated. The field is therefore classified as STATIC-KNOWN.

Header Checksum

The header checksum protects individual hops from processing a corrupted header. When almost all IP header information is compressed away, there is no need to have this additional checksum; instead it can be regenerate at the decompressor side. The field is therefore classified as INFERRED.

Source and Destination addresses

These fields are part of the definition of a stream and must thus be constant for all packets in the stream. The fields are therefore classified as STATIC-DEF.

Summarizing the bits corresponding to the classes gives:

Class	Size (octets)
INFERRED	4
STATIC	1 bit
STATIC-DEF	8
STATIC-KNOWN	3 +7 bits
CHANGING	4

A.1.3. UDP header fields

Field	Size (bits)	Class
Source Port	16	STATIC-DEF
Destination Port	16	STATIC-DEF

Length	16	INFERRED
Checksum	16	CHANGING

Source and Destination ports

These fields are part of the definition of a stream and must thus be constant for all packets in the stream. The fields are therefore classified as STATIC-DEF.

Length

This field is redundant and is therefore classified as INFERRED.

Summarizing the bits corresponding to the classes gives:

Class	Size (octets)
INFERRED	2
STATIC-DEF	4
CHANGING	2

A.1.4. RTP header fields

Field	Size (bits)	Class
Version	2	STATIC-KNOWN
Padding	1	STATIC
Extension	1	STATIC
CSRC Counter	4	CHANGING
Marker	1	CHANGING
Payload Type	7	CHANGING
Sequence Number	16	CHANGING

```
| Timestamp          |      32       |    CHANGING    |
| SSRC               |      32       |   STATIC-DEF   |
| CSRC               |   0(-480)     |    CHANGING    |
+------------------+---------------+----------------+
```

Version

There exists only one working RTP version and that is version 2. The field is therefore classified as STATIC-KNOWN.

Padding

The use of this field depends on the application, but when payload padding is used it is likely to be present in all packets. The field is therefore classified as STATIC.

Extension

If RTP extensions is used by the application, it is likely to be an extension present in all packets (but use of extensions is very uncommon). However, for safety's sake this field is classified as STATIC and not STATIC-KNOWN.

SSRC

This field is part of the definition of a stream and must thus be constant for all packets in the stream. The field is therefore classified as STATIC-DEF.

Summarizing the bits corresponding to the classes gives:

Class	Size (octets)
STATIC	2 bits
STATIC-DEF	4
STATIC-KNOWN	2 bits
CHANGING	7.5(-67.5)

A.1.5. Summary for IP/UDP/RTP

If we summarize this for IP/UDP/RTP we get:

```
+------------------+---------------+----------------+
```

Class \ IP ver	IPv6 (octets)	IPv4 (octets)
INFERRED	4	6
STATIC	2 bits	3 bits
STATIC-DEF	42.5	18
STATIC-KNOWN	1 +6 bits	4 +1 bit
CHANGING	11.5(-71.5)	13.5(-73.5)
Total	60(-120)	40(-100)

A.2. Analysis of change patterns of header fields

To design suitable mechanisms for efficient compression of all header fields, their change patterns must be analyzed. For this reason, an extended classification is done based on the general classification in A.1, considering the fields which were labeled CHANGING in that classification. Different applications will use the fields in different ways, which may affect their behavior. When this is the case, typical behavior for conversational audio and video will be discussed.

The CHANGING fields are separated into five different subclasses:

STATIC — These are fields that were classified as CHANGING on a general basis, but are classified as STATIC here due to certain additional assumptions.

SEMISTATIC — These fields are STATIC most of the time. However, occasionally the value changes but reverts to its original value after a known number of packets.

RARELY-CHANGING (RC) These are fields that change their values occasionally and then keep their new values.

ALTERNATING These fields alternate between a small number of different values.

IRREGULAR These, finally, are the fields for which no useful change pattern can be identified.

To further expand the classification possibilities without increasing complexity, the classification can be done either according to the values of the field and/or according to the values of the deltas for the field.

When the classification is done, other details are also stated regarding possible additional knowledge about the field values and/or field deltas, according to the classification. For fields classified as STATIC or SEMISTATIC, the case could be that the value of the field is not only STATIC but also well KNOWN a priori (two states for SEMISTATIC fields). For fields with non-irregular change behavior, it could be known that changes usually are within a LIMITED range compared to the maximal change for the field. For other fields, the values are completely UNKNOWN.

Table A.1 classifies all the CHANGING fields based on their expected change patterns, especially for conversational audio and video.

Field		Value/Delta	Class	Knowledge
IPv4 Id:	Sequential	Delta	STATIC	KNOWN
	Seq. jump	Delta	RC	LIMITED
	Random	Value	IRREGULAR	UNKNOWN
IP TOS / Tr. Class		Value	RC	UNKNOWN
IP TTL / Hop Limit		Value	ALTERNATING	LIMITED
UDP Checksum:	Disabled	Value	STATIC	KNOWN
	Enabled	Value	IRREGULAR	UNKNOWN
RTP CSRC Count:	No mix	Value	STATIC	KNOWN
	Mixed	Value	RC	LIMITED
RTP Marker		Value	SEMISTATIC	KNOWN/KNOWN
RTP Payload Type		Value	RC	UNKNOWN
RTP Sequence Number		Delta	STATIC	KNOWN

RTP Timestamp		Delta	RC	LIMITED
RTP CSRC List:	No mix	-	-	-
	Mixed	Value	RC	UNKNOWN

Table A.1 : Classification of CHANGING header fields

The following subsections discuss the various header fields in detail. Note that table A.1 and the discussions below do not consider changes caused by loss or reordering before the compression point.

A.2.1. IPv4 Identification

The Identification field (IP ID) of the IPv4 header is there to identify which fragments constitute a datagram when reassembling fragmented datagrams. The IPv4 specification does not specify exactly how this field is to be assigned values, only that each packet should get an IP ID that is unique for the source-destination pair and protocol for the time the datagram (or any of its fragments) could be alive in the network. This means that assignment of IP ID values can be done in various ways, which we have separated into three classes.

Sequential

This assignment policy keeps a separate counter for each outgoing packet stream and thus the IP ID value will increment by one for each packet in the stream. Therefore, the delta value of the field is constant and well known a priori. When RTP is used on top of UDP and IP, the IP ID value follows the RTP sequence number. This assignment policy is the most desirable for header compression purposes but its usage is not as common as it should be. The reason is that it can be realized only if UDP and IP are implemented together so that UDP, which separates packet streams by the port identification, can make IP use separate ID counters for each packet stream.

Sequential jump

This is the most common assignment policy in today's IP stacks. The difference from the sequential method is that only one counter is used for all connections. When the sender is running more than one packet stream simultaneously, the IP ID can increase by more than one. The IP ID values will be much more

predictable and require less bits to transfer than random values, and the packet-to-packet increment (determined by the number of active outgoing packet streams and sending frequencies) will usually be limited.

Random

Some IP stacks assign IP ID values using a pseudo-random number generator. There is thus no correlation between the ID values of subsequent datagrams. Therefore there is no way to predict the IP ID value for the next datagram. For header compression purposes, this means that the IP ID field needs to be sent uncompressed with each datagram, resulting in two extra octets of header. IP stacks in cellular terminals SHOULD NOT use this IP ID assignment policy.

It should be noted that the ID is an IPv4 mechanism and is therefore not needed at all in IPv6 profiles. For IPv4 the ID could be handled in three different ways. Firstly, we have the inefficient but

reliable solution where the ID field is sent as-is in all packets, increasing the compressed headers with two octets. This is the best way to handle the ID field if the sender uses random assignment of the ID field. Secondly, there can be solutions with more flexible mechanisms requiring less bits for the ID handling as long as sequential jump assignment is used. Such solutions will probably require even more bits if random assignment is used by the sender. Knowledge about the sender's assignment policy could therefore be useful when choosing between the two solutions above. Finally, even for IPv4, header compression could be designed without any additional information for the ID field included in compressed headers. To use such schemes, it must be known that the sender makes use of the pure sequential assignment policy for the ID field. That might not be possible to know, which implies that the applicability of such solutions is very uncertain. However, designers of IPv4 stacks for cellular terminals SHOULD use the sequential policy.

A.2.2. IP Traffic-Class / Type-Of-Service

The Traffic-Class (IPv6) or Type-Of-Service (IPv4) field is expected to be constant during the lifetime of a packet stream or to change relatively seldom.

A.2.3. IP Hop-Limit / Time-To-Live

The Hop-Limit (IPv6) or Time-To-Live (IPv4) field is expected to be constant during the lifetime of a packet stream or to alternate between a limited number of values due to route changes.

A.2.4. UDP Checksum

The UDP checksum is optional. If disabled, its value is constantly zero and could be compressed away. If enabled, its value depends on the payload, which for compression purposes is equivalent to it

changing randomly with every packet.

A.2.5. RTP CSRC Counter

This is a counter indicating the number of CSRC items present in the CSRC list. This number is expected to be almost constant on a packet-to-packet basis and change by small amount. As long as no RTP mixer is used, the value of this field is zero.

A.2.6. RTP Marker

For audio the marker bit should be set only in the first packet of a talkspurt while for video it should be set in the last packet of every picture. This means that in both cases the RTP marker is classified as SEMISTATIC with well-known values for both states.

A.2.7. RTP Payload Type

Changes of the RTP payload type within a packet stream are expected to be rare. Applications could adapt to congestion by changing payload type and/or frame sizes, but that is not expected to happen frequently.

A.2.8. RTP Sequence Number

The RTP sequence number will be incremented by one for each packet sent.

A.2.9. RTP Timestamp

In the audio case:

As long as there are no pauses in the audio stream, the RTP timestamp will be incremented by a constant delta, corresponding to the number of samples in the speech frame. It will thus mostly follow the RTP sequence number. When there has been a silent period and a new talkspurt begins, the timestamp will jump in proportion to the length of the silent period. However, the increment will probably be within a relatively limited range.

In the video case:

The timestamp change between two consecutive packets will either be zero or increase by a multiple of a fixed value corresponding to the picture clock frequency. The timestamp can also decrease by a multiple of the fixed value if B-pictures are used. The delta interval, expressed as a multiple of the picture clock frequency, is in most cases very limited.

A.2.10. RTP Contributing Sources (CSRC)

The participants in a session, which are identified by the CSRC fields, are expected to be almost the same on a packet-to-packet basis with relatively few additions or removals. As long as RTP mixers are not used, no CSRC fields are present at all.

A.3. Header compression strategies

This section elaborates on what has been done in previous sections. Based in the classifications, recommendations are given on how to handle the various fields in the header compression process. Seven different actions are possible and these are listed together with the fields to which each action applies.

A.3.1. Do not send at all

The fields that have well known values a priori do not have to be sent at all. These are:

- IP Version
- IPv6 Payload Length
- IPv6 Next Header
- IPv4 Header Length
- IPv4 Reserved Flag
- IPv4 Last Fragment Flag
- IPv4 Fragment Offset
- IPv4 Protocol
- UDP Checksum (if disabled)
- RTP Version

A.3.2. Transmit only initially

The fields that are constant throughout the lifetime of the packet stream have to be transmitted and correctly delivered to the decompressor only once. These are:

- IP Source Address
- IP Destination Address
- IPv6 Flow Label
- IPv4 May Fragment Flag
- UDP Source Port
- UDP Destination Port
- RTP Padding Flag
- RTP Extension Flag
- RTP SSRC

A.3.3. Transmit initially, but be prepared to update occasionally

The fields that are changing only occasionally must be transmitted initially but there must also be a way to update these fields with

new values if they change. These fields are:

- IPv6 Traffic Class
- IPv6 Hop Limit

- IPv4 Type Of Service (TOS)
- IPv4 Time To Live (TTL)
- RTP CSRC Counter
- RTP Payload Type
- RTP CSRC List

A.3.4. Be prepared to update or send as-is frequently

For fields that normally are either constant or whose values can be deduced from some other field but frequently diverge from that behavior, there must be an efficient way to update the field value or send it as-is in some packets. Those fields are:

- IPv4 Identification (if not sequentially assigned)
- RTP Marker
- RTP Timestamp

A.3.5. Guarantee continuous robustness

Fields that behave like a counter with a fixed delta for ALL packets, the only requirement on the transmission encoding is that packet losses between compressor and decompressor must be tolerable. If more than one such field exists, all these can be communicated together. Such fields can also be used to interpret the values for fields listed in the previous section. Fields that have this counter behavior are:

- IPv4 Identification (if sequentially assigned)
- RTP Sequence Number

A.3.6. Transmit as-is in all packets

Fields that have completely random values for each packet must be included as-is in all compressed headers. Those fields are:

- IPv4 Identification (if randomly assigned)
- UDP Checksum (if enabled)

A.3.7. Establish and be prepared to update delta

Finally, there is a field that is usually increasing by a fixed delta and is correlated to another field. For this field it would make sense to make that delta part of the context state. The delta must then be possible to initiate and update in the same way as the fields listed in A.3.3. The field to which this applies is:

- RTP Timestamp

Appendix COMPLIST

Editor's Note: This is too long and not quite in style to be added to encoding section. Also, I believe this is too complicated.

1. Compression of Item List

An Item List specifies a set of items that is order sensitive. A generic structure of an item list is as follows:

```
             +--------+--------+--------------------+
  Item list: | item 1 | item 2 | ......  | item n |
             +--------+--------+--------------------+
```

The items on an item list may or may not have the same structure. An example of the first case is the CSRC list in the RTP header or the Address list in Type 0 IPv6 Routing Header. An example of the second case is IPv6 extension headers. Note that an item itself could be a list.

The compression of a current item list (curr_list) is based on a reference item list (ref_list) that is previously sent by the compressor and received by the decompressor. The difference between the curr_list and ref_list is encoded and sent in the compressed list. The reference may be chosen by various means. One approach is based on acknowledgement, i.e., the compressor chooses the latest item list that is acknowledged by the decompressor as the reference. The decompressor retrieves the reference item list, based on which items are to be decompressed in the new item list. To identify the reference used for the compression, a reference identifier (ref_id) needs to be carried inside the compressed list. An example of ref id is RTP sequence number.

1.1. Item Compression

A given item in curr_list can be classified as belonging to one of the following transformation cases.

* Transformation Case A: The given item is also in the ref_list, and the content of these two items is the same, while the position in the list may or may not have changed.

* Transformation Case B: There is an item in the ref_list with a similar structure, but the contents of these two items are not identical. The position in the list may or may not be the same.

* Transformation Case C: The given item in the curr_list is not in the ref_list.

For the given item in curr_list, the compressor determines which transformation case applies. Depending on the transformation case, the correspondent encoding technique is used.

1.2. List Compression

Seven encoding schemes are defined below. To identify the encoding scheme used for the list compression, an encoding type field (ET) is included in the compressed list.

1.2.1. Generic Scheme

The generic encoding scheme addresses the case where the items belong to a mixture of transformation cases. For example, item 1 in curr_list belongs to transformation case B, item 2 belongs to transformation case C, while item 3 belongs to transformation case A.

Using this scheme, the structure of a compressed item list is defined as follows.

```
      compressed list:
      +----------+--------+-------------+----------+-----+----------
+
      | ET = 000 | ref_id | c_item count | c_item 1 | ... | c_item n
|
      +----------+--------+-------------+----------+-----+----------
+
```

* The encoding type (ET) is "000".

* The ref_id is defined at the beginning of section 1. One example is the RTP sequence number.

* The c_item count specifies the number of c_items in the c_item list (c_item 1, ..., c_item n). The length of c_item count is defined based on the context of the application.

* Each c_item (c_item i) corresponds to one of the item (item i) in the curr_list; thus the order of the c_items represents the order of the items in the curr_list. The structure of c_item is defined as follows.

Each c_item consists of C_item Code (CC) and C_item Data.

```
             +-------------+-------------+
     c_item: | c_item code | c_item data |
             +-------------+-------------+
```

For different c_item codes, the content of the c_item data is different. Three types of c_item codes and their respective c_item data are defined as follow.

* C_item Code "0" is used for an item classified as belonging to transformation case A. The structure of the c_item is as follows.

```
                 +-----------+------+
        c_item:  | CC = "0"  | pos  |
                 +-----------+------+
```

- The "pos" field indicates the position of the item in the ref_list.

Note that "pos" starts from 0, i.e., the position of the first item in the list is 0. Since the "pos" field in the c_item data field indicates the position of the item in the ref_list, the length of "pos" field depends on the actual number of items in the ref_list. Assuming that there are 'k' items in the ref_list, then the number of bits used for "pos" field is ceiling(log2(k)). Since k is known to both the compressor and decompresser, the length of "pos" field doesn't need to be carried in the compressed list.

When the decompressor receives this c_item, it retrieves the item at position "pos" in the ref_list, and copies it into the curr_list.

* C_item Code "10" is used for an item classified as belonging to transformation case B. The structure of the c_item is as follows.

```
                 +-----------+------+:::::::::::::::::::+------------------+
        c_item:  | CC = "10" | pos  | type-specific data | compressed data |
                 +-----------+------+:::::::::::::::::::+------------------+
```

- The "pos" field is defined as above.

- The "compressed data" field carries compressed value of the item in the curr_list, using the item at position "pos" in the ref_list as the reference. The compression technique is dependent on the item and should be predefined.

- The optional "type-specific data" field contains additional information used to reconstruct the item list. The presence and the content of the "type-specific data" depend on the item and should be predefined.

When the decompressor receives this c_item, it retrieves the item at the position "pos" in the ref_list and uses it as the reference to decompress the "compressed data". If the item itself is a list, the compression scheme described here applies to compress the item.

* C_item Code "11" is used for an item classified as belonging to transformation case C or transformation case B in the case that the length of the compressed value of an item is even larger than the length of the uncompressed value due to too many changes. The structure of the c_item is as follows.

```
                 +-----------+:::::::::::::::::::+------------------
```

```
--+
        c_item: | CC = "11" | type-specific data | uncompressed data
|
                 +------------:::::::::::::::::::::+-----------------
-+
```

- The optional "type-specific data" field is defined as above.

- The "uncompressed data" field contains the original value of the item in the curr_list.

When the decompressor receives this c_item, it copies the uncompressed data field into curr_list.

1.2.2. Insertion Only Scheme

This scheme only addresses the case where all the items are classified as belonging to either transformation case A or C. The structure of the compressed item list is as follows.

```
                   +-----------+-------+------------------+-------+
compressed list:   | ET = 001  | ref_id | u_item count    | aft   |
                   +-----------+-------+------------------+-------+
+---------------------+-------------+----+-----------+
 insertion order      | u_item 1    |....| u_item m  |
+---------------------+-------------+----+-----------+
```

* The encoding type (ET) is "001".

* The ref_id is defined as before.

* The u_item count field carries the number of u_items in the ... u_item list (u_item 1,., u_item m). The length of u_item ... is defined based on the context of the application.

* Each u_item carries the uncompressed value of the new item in the curr_list when comparing it with the ref list.

```
          +---------------------+
u_item:   | uncompressed data   |
          +---------------------+
```


* The aft field specifies the format used in insertion order field.
 0 - bit mask format
 1 - position list format

* The insertion order field specifies the positions of the inserted items. Two formats can be used.

** Bit Mask Format: In this format, a bit mask is include.

To construct the bit mask and the u_item list, the following steps are taken.

*** A list of '0' and an empty u_item list are generated as the starting point. The number of '0's in the '0' list equals the number of items in the ref_list. The position of each '0' in the '0' list corresponds to the position of an item in the ref_list, i.e., the i-th '0' in the '0' list corresponds to the i-th item in the ref_list.

*** Comparing the curr_list with the ref_list, if a new item is inserted between the i-th item and the (i+1)-th item in the ref_list, a '1' is inserted between the i-th '0' and (i+1)-th '0' in the original '0' list. The u_item that carries the new item should be added to the end of u_item list. This procedure is repeated until all the added items have been processed.

Assuming the number of items in the ref_list is k, and the number of new items is m, then the length of the bit mask is k+m. Since k is known to both the compressor and decompressor, and m is carried in "u_item count" field, the length of the bit mask can be obtained by the compressor and decompressor and doesn't need to be carried in the compressed item list.

When the decompressor receives the bit mask, it scans from left to right. When a '0' is observed, the decompressor copies the corresponding item in the ref_list into the curr_list; when an '1' is observed, the decompressor adds the correspondent u_item into the curr_list.

** Position List Format: In this format, a list of position fields is carried. The structure of this format is as follows.

```
               +--------+--------+--------+
position list: | pos 1  | ....   | pos m  |
               +--------+--------+--------+
```

The value of Pos i specifies the position of the item in the ref_list, before which u_item i should be inserted. If two or more u_items have the same position value, they are inserted in the order they appear. A u_item is inserted after a previous inserted u_item.

When the decompressor receives position list, it first retrieves all the items in the ref_list. Then for each u_item, it inserts it into the ref_list at the position indicated in the corresponding pos field in the position list.

Note that the number of bits used for pos field is ceiling(log2(k+1)), where k is the number of items in the ref_list. Pos field = "k" means that the correspondent item should be inserted to the end of the list.

Assuming the number of new items is m, then the total length of the position list is m * ceiling(log2(k+1)).

The selection of these formats depends on the encoding efficiency. In the case that the number of item in the ref_list is large and only few items are inserted, the position list format is favorable; otherwise the bit mask format is more efficient.

1.2.3. Removal Only Scheme

This scheme only addresses the case where certain items exist in the ref_list but not in the curr_list. The structure of the compressed item list is as follows.

```
                   +----------+--------+-----+---------------+
compressed list:   | ET = 010 | ref_id | rft | removal order |
                   +----------+--------+-----+---------------+
```

* The encoding scheme type (ET) is "010".

* The ref_id is defined as before.

* The rft field specifies the format used in removal order field.
  0 - bit mask format
  1 - position list format

* The removal order field specifies the positions of the items to be removed. Two formats can be used.

** Bit mask format: In this format, a removal bit mask is included. A '1' in the i-th bit in the removal bit mask means that the i-th item in the ref_list is not included in the curr_list, while a '0' means it is still present in the curr_list. Assuming the number of items in the ref_list is k, then the length of the removal bit mask is k. Since k is known to both the compressor and decompressor, the length of the bit mask doesn't need to be carried in the compressed list.

** Position list format: In this format, a list of position fields as well as a count field is included. The structure of this format is as follows.

```
                +-------+-------+-----+-------+
position list:  | count | pos 1 | ... | pos m |
                +-------+-------+-----+-------+
```

- The count field carries the number of pos fields in the position list. Count field = '0' has the special meaning that all the items are removed and no pos field is included. Assuming the number of items in the ref_list is k, then the length of count field is ceiling(log2(k)).

- Each pos field carries the position of an item in the ref_list, which no longer exists in the curr_list. The length of the pos field is ceiling(log2(k)).

The length of the position list is (m + 1) * ceiling(log2(k)).

The selection between these two formats depends on the encoding efficiency. In the case that the number of items in the

ref_list is small and/or the number of items removed from the ref_list is large, the bit mask format is favorable; otherwise the position list format is more efficient.

1.2.4. Insertion and Removal Only Scheme

This scheme accommodates all the transformation cases addressed in insertion only and removal only schemes. The structure of the compressed item list is as follows.

```
                   +----------+--------+------------------+-----+
compressed list: | ET = 011 | ref_id | u_item count | cft |
                   +----------+--------+------------------+-----+

  ---------------+-----+-----------------+----------+-----+--------
--+
   removal order | aft | insertion order | u_item 1 | ... | u_item
m |
  ---------------+-----+-----------------+----------+-----+--------
--+
```

* The encoding scheme type (ET) is "011".

* The ref_id is defined as before.

* The remaining fields are defined in insertion only scheme and removal only scheme. Unlike the insertion order field in the insertion only case, the insertion order field in this scheme is based on the items left in the ref_list after removal has been

processed. When the decompressor receives such compressed list, it applies the removal before the insertion.

1.2.5. Content Change Only Scheme

This scheme only addresses the case where the number of items in the list and the ordering are not changed, but the content of one or more item is changed. The structure of the compressed item list is as follows.

```
  compressed list:
  +----------+--------+-----+--------------+-----------+-----+------
--------+
  | ET = 100 | ref_id | cft | change order | uc_item 1 | ... |
uc_item m |
  +----------+--------+-----+--------------+-----------+-----+------
--------+
```

* The encoding type (ET) is "100".

* The ref_id is defined as before.

* The cft field specifies the format used in change order field.
  0 - bit mask format
  1 - position list format

* The change order field specifies the position of the items whose content is changed. Two formats can be used.

** Bit mask format: In this format, a change bit mask is included. A '1' in the i-th bit in the change bit mask means that the i-th item in the ref_list is not identical to the i-th item in the curr_list, while a '0' means they are the same. Assuming the number of items in the ref_list is k, then the length of the change bit mask equals k. Since k is known to both the compressor and the decompressor, it doesn't need to be sent in the compressed list.

** Position list format: In this format, a list of position fields as well as a count field is included. The structure of this format is as follows.

```
                  +-------+-------+-----+-------+
   position list: | count | pos 1 |.....| pos m |
                  +-------+-------+-----+-------+
```

- The count field carries the number of pos fields in the position list. Value '0' in count field has the special meaning that all the items are changed and no pos field is included. Assuming the number of items in the ref_list is k, then the length of count field is ceiling(log2(k)).

- Each pos field carries the position of an item in the ref_list, whose value is not identical with the item at the same position in the curr_list.

The length of position list is (m + 1) * ceiling(log2(k)).

The selection between these two formats depends on the encoding efficiency. In the case that the number of items that have content change is small, the Bit mask format is favorable; otherwise the position list format is more efficient.

* Each uc_item in the uc_item list corresponds to the item whose content is changed when comparing it with the item at the same position in the ref_list. Their positions in curr_list is indicated in the change order field. When position list format is used in change order field and the count field is '0', the order of the uc_item is the same as the item order in ref_list. The structure of uc_item is as follows.

```
             +---+----------------------------------+
    uc_item: | C | compressed or uncompressed data |
             +---+----------------------------------+
```

C bit specifies the format of the compressed or uncompressed data field. A '0' in C bit indicates the uncompressed value of the item is sent, while a '1' indicates the compressed value of the item is carried. The item in the curr_list is compressed using the item at the same position in ref_list as the reference. The compression technique is dependent on the item and should be predefined.

1.2.6. Insertion and Content Change Only Scheme

This scheme only addresses the case where a) all the items in

the ref_list are also in the curr_list, 2) new items are added into the curr_list, 3) the relative order of the items that are in both ref_list and curr_list remains the same, and 4) the content of one or more of these items have been changed. The structure of the compressed item list is as follows.

```
compressed list:
+----------+--------+-----+--------------+-----------+-----+----------+
| ET = 101 | ref_id | cft | change order | uc_item 1 |.....| uc_item m1 |
+----------+--------+-----+--------------+-----------+-----+----------+
+-----+-----------------+----------+-----+-----------+
 aft  | insertion order | u_item 1 |.....| u_item m2 |
+-----+-----------------+----------+-----+-----------+
```

* The encoding type (ET) is "101".

* The ref_id is defined at the beginning of section 1.

* The remaining fields are defined in insertion only scheme and content change only scheme. The change order field in this scheme is based on the items in the ref_list and the items in the curr_list, excluding the new inserted items. When the decompressor receives such a compressed list, it applies the content change before the insertion.

1.2.7. Removal and Content Change Only Scheme

This scheme only addresses the case where 1) some items in the ref_list are not in the curr_list, and 2) the content of one or more items that are in both ref_list and curr_list is changed, but the relative order of these items remains the same. The structure of the compressed item list is as follows.

```
compressed list:
+----------+--------+-----+---------------+-----+--------------+
| ET = 110 | ref_id | rft | removal order | cft | change order |
+----------+--------+-----+---------------+-----+--------------+
+-----------+-----+-----------+
 uc_item 1  |.....| uc_item m |
+-----------+-----+-----------+
```

* The encoding type (ET) is "110".

* The ref_id is defined at the beginning of section 1.

* The remaining fields are defined in the removal only scheme and content change only scheme. The change order field in this scheme is based on the items in the curr_list and the items in the ref_list after the removal is processed. When the decompressor receives such a compressed list, it applies the removal before the content change.

2. Examples

The examples below illustrate the operation of item list compression under various transformation cases. We assume no type-specific data is needed for the decompression.

Example 1. Insertion and Removal only

Assuming the items and the order of these items in the curr_list and ref_list are as follow.

curr_list: A, B, C, D

ref_list: B, C, X

Comparing curr_list with ref_list, A and D are added into the list and X is removed from the list. No change happens to the order and the content for B and C. The format defined in the insertion and removal only scheme can be used.

The compressed item list format for this case is as follow.

```
                  +-----------+---------+-------------------+
compressed list:  | ET = 011  | ref_id  | u_item count | rft |
                  +-----------+---------+-------------------+

-------------------+-----+--------------------+------------------+--------
-----+
 removal order | aft | insertion order | u_item for A | u_item
for D |
-------------------+-----+--------------------+------------------+--------
-----+
```

* ref_id is defined in section 1.

* The u_item count equals 2.

* Assuming that the bit mask format is used for removal order, then rft equals '0'.

* The removal order is "001" where the bits from left to right correspond to B, C, X respectively. The '1' corresponds to X and indicates that X is removed.

* Assuming that position list format is used for the insertion order, then aft equals '1'.

* The insertion order is "0010". The first 2 bits "00" indicate item A is added before the item at position "00" in the ref_list, which is B. The following 2 bits "10" indicate item D is added before position "10" in the ref_list after the removal process, which is after item B.

* The u_item for A and u_item for D carry uncompressed A and D respectively.

Example 2. Insertion, Removal, Change of Content and Reordering

Assuming the items and the order of these items in the curr_list and ref_list are as follow.

curr_list: A, C, B', D

ref_list: B, C, X

Comparing curr_list with ref_list, A and D are added into the list and X is removed from the list. The order of B and C is changed and the content of B is changed as well. The generic item list compression format is used to carry the change.

```
                  +----------+---------+-----------------+
compressed list:  | ET = 000 | ref_id  | c_item count    |
                  +----------+---------+-----------------+
-----------+------------+-----------+-----------+
 c_item A  |  c_item C  |  c_item B |  c_item D |
-----------+------------+-----------+-----------+
```

* The c_item count equals 4.

```
             +----+------------------------+
* c_item A:  | 11 | A (uncompressed value) |
             +----+------------------------+

             +---+----+
* c_item C:  | 0 | 01 |
             +---+----+
```

"01" represents the position of C in the ref_list.

```
             +----+----+---------------+
* c_item B:  | 10 | 00 | compressed B' |
             +----+----+---------------+
```

"00" represents the position of B in the ref_list. Compressed B' carries the compressed value of B', using B in the ref_list as the reference.

```
             +----+------------------------+
* c_item D:  | 11 | D (uncompressed value) |
             +----+------------------------+
```

3. Optimization

The above description assumed that a given item in the curr_list can be uniquely classified as belonging to one transformation case. In reality, there are cases where there is more than one way to do the classification. An example is given as follows.

For a given item (item X) in the curr_list, there is no item in the ref_list which has an identical content, but one item (item Y) in the ref_list has a similar structure to item X, therefore, item X can be classified as belonging to either transformation case B or transformation case C. An example of this type of list is the CSRC

list in RTP header.

The compressor can decide to use the transformation case, for which the encoding scheme will yield the highest compression efficiency. For example, in the above example, let's assume item X consists of L_x bits.

* If item X is classified as belonging to transformation case A and c_item code "11" is used, the c_item for item X consists of (L_x + 2) bits.

* If item X is classified as belonging to transformation case B and c_item code "10" is used, under the assumption that the length of the pos field is L_pos and the length of compressed item X when using item Y as the reference is D_xy, then the c_item for item X consists of (L_pos + D_xy + 2) bits.

Thus, if (L_pos + D_xy) is larger than L_x, then transformation case B is applied; otherwise transformation case A is applied.

Appendix E. Encoding Examples

E.1. Basic VLE

The examples below illustrate the operation of VLE under various scenarios. The field values used in the examples could correspond to any fields that we wish to compress. The examples illustrate the scenario where the compressed field has resolution of one bit.

Example 1: Normal operation (no packet loss prior to compressor, no reodering prior to compressor).

Suppose packets with header fields 279, 280, 281, 282, and 283 have been sent, and 279 and 283 are fields of potential reference packets.

The current VLE window is {279, 283}.

and a packet with field value = 284 is received next, VLE computes the following values

New Value	VMax	VMin	r	# LSBs
284	283	279	max(|284-279|,|284-283|)=5	4

The window is unmodified if we assuming the new packet (284) is not a potential reference. The field is encoded using 4 bits in this case, and the actual encoded value is the 4 least significant bits of 284 (100011100) which = 1100.

Example 2: Packet Loss prior to compressor.

Suppose packets with header fields 279, 280, 281, 282, and 283 have been sent, and 279 and 283 are fields of potential reference packets such that the VSW is again {279, 283}.

If a packet with field value = 290 is received next, VLE computes the following values

New Value	VMax	VMin	r	# LSBs
290	283	279	max(|290-283|,|290-279|)=11	5

So the field is encoded using 5 bits. Actual encoded value is the 5 LSBs of 290 (100100010) which = 00010.

If we assume the new value is a potential reference, the new VSW is {279, 283, 290}.

Example 3: Packet Misordering prior to compressor.

Suppose packets with header fields 279, 280, 281, 282, and 283 have been sent, and 279 and 283 are fields of potential reference packets such that the VSW is again {279, 283}.

If a packet with field value = 278 is received next, VLE computes the following values

New Value	VMax	VMin	r	# LSBs
278	283	279	max{|278-283|,|278-279|}=5	4

So the field is encoded using 4 bits. Actual encoded value is the 4 LSBs of 278 (10010110) which = 0110.

If we assume the new value is a potential reference, the new VSW is {283, 280, 278}.

In any case, the VLE encoded fields must be accompanied by some bits in order to identify the different possible encoded field sizes. Sizes of this bit field can vary depending on the number of different sizes one wishes to allow. The approach in ACE is to allow only a few different sizes for byte-aligned header formats. Huffman coding of the length is used to achieve some additional efficiency, based on the expected frequency of needing the different sizes. 1 or 2 additional bits are actually sent in the ACE compressed header.

The decompressor behavior in all the example cases is the same- it uses as a reference a specific decompressed header field value. The header to use might be indicated by the presence of a checksum in the compressed header packet, or by other means. They must by definition be one of the values in the compressor's window.

For example let's assume that the last correctly decompressed packet which qualifies as a reference was the packet with header field = 291. Now suppose the encoded field value of 303 (100011111) is received and = 01111. The two values closest values to 291 which have LSBs = 01111 are 271 and 303. 303 is closest, therefore it is correctly selected as the uncompressed field value.

E.2. Timer-Based VLE

As a an example of operation, consider the case of a voice codec (20 mS), such that TS_stride = 160. Assume T_current and p_TS_current are 357 and 351, respectively, and that we have sliding window TSW which contains the following values 4 entries:

j	T_j	p_TS_j
1	9	7
2	8	6

j	T_j	p_TS_j
3	7	4
4	3	1

j above is the packet number.

```
   in this case we have

   Network_jitter(1)=|(357-9)-(351-7)|=4  (80 mS Network Jitter)
   Network_jitter(2)=|(357-8)-(351-6)|=4  (80 mS Network Jitter)
   Network_jitter(3)=|(357-7)-(351-4)|=3  (60 mS Network Jitter)
   Network_jitter(4)=|(357-3)-(351-1)|=4  (80 mS Network Jitter)

   So Max_Network_Jitter = 4.

   We assume a maximum CD-CC jitter of 2 (40 mS); the total jitter to
be handled in this case is then

         J = 4 + 2 + 2 = 8 packets (160 mS)

   and k = 5 bits (since 2 * 8 + 1 < 2^5).  The compressor sends the
5 LSBs of p_TS_current to the decompressor (351 = 101011111, so the
encoded TS value = 11111).

   When the decompressor receives this value, it first attempts to
estimate the timestamp by computing the time difference between the
last reference established and the current packet

          T_current - T_ref, where T_ref is the value of the wall
clock time at which the reference headers was received by the
decompressor


   That value is added to p_TS_ref, the packed RTP TS of the
reference header, to get the estimate.

   Assume that at the decompressor packet #3 is used as the
reference:


         - T_current = 359
         - T_ref = 7
         - p_TS_ref = 4

   Note:

   T_current is picked here as any value; the difference between it
and T_ref represents the length of the silence interval as observed
at the decompressor.  Then:

         T_current - T_ref = 359 - 7 = 352
         p_TS_current(estimate) = 352 + 4 = 356
```



```
   The decompressor searches for the closest value to 356 which has,
in this case, LSBs = 11111.  The value in this case is 351, the
original p_TS.


   If instead the compressor were to send the timestamp jump as
simply the difference in consecutive packed RTP Timestamps, that
```

value would be

p_TS_current - p_TS_ref = 351-4 = 347 = 101011011

So over twice as many bits would be sent for a silence interval of

347 (20 mS) = 6.94 seconds

Due to basic conversational real-time requirements, the cumulative jitter in normal operation is expected to be at most only a few times T stride for voice. For this reason, the FO payload formats in section 4.3 are optimized (in terms of representing different k-length encoded TS values) for the case of k=4 (handles up to 16 discrepencies in the timestamp). The remaining formats allow a wide range of jitter conditions (outside of just voice) to be handled as well.

This Internet-Draft expires December 27, 2000.

Ref. 2

Network Working Group
INTERNET-DRAFT
Expires: August 2005

Ghyslain Pelletier, Editor, Ericsson
Lars-Erik Jonsson, Ericsson
Mark A West, Siemens/Roke Manor
Carsten Bormann, TZI/Uni Bremen
Kristofer Sandlund, Effnet
February 21, 2005

RObust Header Compression (ROHC): A Profile for TCP/IP (ROHC-TCP)

<draft-ietf-rohc-tcp-09.txt>

Status of this memo

This document is a submission of the IETF ROHC WG. Comments should be directed to the ROHC WG mailing list, rohc@ietf.org.

Abstract

This document specifies a ROHC (Robust Header Compression) profile for compression of TCP/IP packets. The profile, called ROHC-TCP, is a robust header compression scheme for TCP/IP that provides improved compression efficiency and enhanced capabilities for compression of various header fields including TCP options.

Existing TCP/IP header compression schemes do not work well when used over links with significant error rates and long round-trip times. For many bandwidth-limited links where header compression is essential, such characteristics are common. In addition, existing schemes (RFC 1144 [14], RFC 2507 [21]) have not addressed how to compress TCP options such as SACK (Selective Acknowledgements) (RFC

2018 [20], RFC 2883 [22]) and Timestamps (RFC 1323 [15]).

Table of Contents

ETSI TR 125 925 V3.3.0 (2000-12)

Technical Report

XP-002230388

Universal Mobile Telecommunications System (UMTS); Radio Interface for Broadcast/Multicast Services (3GPP TR 25.925 version 3.3.0 Release 1999)

Reference

RTR/TSGR-0225925UR3

Keywords

UMTS

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - NAF 742 C
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° 7803/88

Important notice

Individual copies of the present document can be downloaded from:
http://www.etsi.org

The present document may be made available in more than one electronic version or in print. In any case of existing or perceived difference in contents between such versions, the reference version is the Portable Document Format (PDF). In case of dispute, the reference shall be the printing on ETSI printers of the PDF version kept on a specific network drive within ETSI Secretariat.

Users of the present document should be aware that the document may be subject to revision or change of status. Information on the current status of this and other ETSI documents is available at http://www.etsi.org/tb/status/

If you find errors in the present document, send your comment to:
editor@etsi.fr

Intellectual Property Rights

IPRs essential or potentially essential to the present document may have been declared to ETSI. The information pertaining to these essential IPRs, if any, is publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (http://www.etsi.org/ipr).

Pursuant to the ETSI IPR Policy, no investigation, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Foreword

This Technical Report (TR) has been produced by the ETSI 3rd Generation Partnership Project (3GPP).

The present document may refer to technical specifications or reports using their 3GPP identities, UMTS identities or GSM identities. These should be interpreted as being references to the corresponding ETSI deliverables.

The cross reference between GSM, UMTS, 3GPP and ETSI identities can be found under www.etsi.org/key .

Contents

Foreword

This Technical Report (TR) has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

1 Scope

The present document shall provide a general overview on radio interface related aspects of broadcast/multicast services. This report covers stage 2 and stage 3 aspects of the radio interface.

This report is organised as follows: clause 4 gives an overview on the broadcast/multicast services and their requirements. Clause 5 provides a common model and describes aspects common to all point-to-multipoint services. Clauses 6 to 10 are devoted to the different broadcast/multicast service categories. Each service specific clause describes the requirements on the interfaces. In these subclauses the impacts on the interface functions and the protocol aspects are described. The present document covers only those items which are in the scope of 3GPP TSG RAN WG 2. Information from Technical Specifications or other documents are provided when it is necessary to understand the requirements described.

Table 1.1: Schedule of the broadcast/multicast services onto the UMTS phases and annual releases

Phase	Release	Broadcast/multicast service
1	1999	Cell Broadcast Service CBS (GSM) Cell Broadcast Service (ANSI-41)

NOTE: A decision to map the services to phases/releases is required for all other broadcast/multicast services.

2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies.

[1] 3GPP TS 22.100: "UMTS Phase 1".

[2] 3GPP TS 22.101: "UMTS Service Principle".

[3] 3GPP TS 22.105: "Services and Service Capabilities".

[4] 3GPP TS 25.301: "Radio Interface Protocol Architecture".

[5] 3GPP TS 25.302: "Services provided by the Physical Layer".

[6] 3GPP TS 25.303: "UE Functions and Interlayer Procedures in Connected Mode".

[7] 3GPP TS 25.304: "UE Procedures in Idle Mode".

[8] 3GPP TS 25.321: "MAC Protocol Specification".

[9] 3GPP TS 25.322: "RLC Protocol Specification".

[10] 3GPP TS 25.331: "RRC Protocol Specification".

[11] 3GPP TS 22.003: "Digital cellular telecommunications system (Phase 2+); Principles of telecommunication services supported by a GSM Public Land Mobile Network (PLMN)".

[12] 3GPP TS 23.060: "General GPRS Service description; Stage 2".

[13] 3GPP TS 23.041: "Technical realization of Cell Broadcast Service (CBS)".

[14] GSM 03.61: "Digital cellular telecommunications system (Phase 2+); Point To Multipoint Multicast Service Description; Stage 2".

[15] 3GPP TS 23.110: "UMTS Access Stratum, Services and Functions".

[16] 3GPP TS 25.324: "Broadcast/Multicast Protocol BMC".

[17] 3GPP TS 24.012: "Short Message Service Cell Broadcast (SMS) Support on the Mobile Radio Interface".

[18] 3GPP TS 25.402: "Synchronization in UTRAN Stage 2".

[19] 3GPP TS 25.419: "Service Area Broadcast Protocol SABP".

[20] TIA/EIA-637-A: "TR45 – Short Message Service for Spread Spectrum Systems".

3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

BCCH Broadcast Control Channel
BCH Broadcast Channel
BMC Broadcast/Multicast Control
BM-IWF Broadcast/Multicast Interworking Function
CB Cell Broadcast
CBS Cell Broadcast Service
CCCH Common Control Channel
CTCH Common Traffic Channel
CTCH-BS Common Traffic Channel Block Set
DRX Discontinuous Reception
FACH Forward Access Channel
IP Internet Protocol
IP-M IP Multicast
MDS Multimedia Distribution Service
PTM Point-to-Multipoint
PTM-G PTM Group Call
PTM-M PTM Multicast
SMS Short Message Service
SMS-CB SMS Cell Broadcast
UE User Equipment
UMTS Universal Mobile Telecommunication System
TB Transport Block
TR Technical Report
TS Technical Specification
TrCH Transport channel
UTRAN UMTS Terrestrial Radio Access Network

4 Overview of Point-to-multipoint Services and Requirements

It is agreed to have service continuity for GSM/GPRS point-to-multipoint services in UMTS ([1] and [2]). This means that the user gets the same service behaviour as he knows it from GSM or GPRS. The services are Cell Broadcast Service [13] and Point-to-multipoint Multicast, Point-to-multipoint Group Call and IP Multicast [14].

Combined with the UMTS service classification given in [2] the service classification shown in Figure 1 could be used as a starting point. The figure contains the view in terms of Radio Access Bearer services and should not be applied for higher layers where other categories of services may exist. Future work may result in changes of Figure 1.

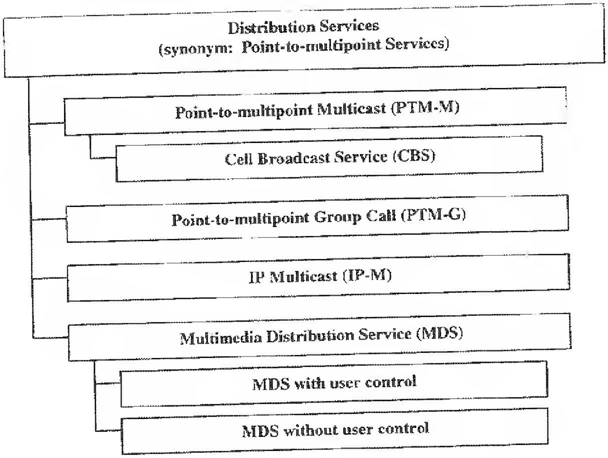

Figure 4.1: Structure of point-to-multipoint services

Table 4.1 gives an overview of broadcast/multicast services as recognised on the radio interface.

Table 4.1: Radio Interface related attributes of broadcast/multicast services [12]

Attributes	CBS (SMS-CB)	PTM Multicast	PTM Group call	IP-multicast
UE modes	Idle Connected			
Logical Channels	CTCH	CTCH	CTCH	CTCH
Necessity of separate control channel	Yes BCCH			
Transport Channels	FACH			
Physical Channels	Secondary CCPCH			
DRX Mode	Yes	Yes	No	Yes
Primary addressing	GEO area	Subscriber group	Subscriber group	Subscriber group
Secondary addressing	---	GEO area	GEO area	---
Present subscribers known	No	No	Yes	Yes
Ciphering	No	No	Yes	Yes
Reliable delivery	No	No	Optional	Yes

5 Common Model

The Common Traffic Channel (CTCH) [4] is provided by the MAC sublayer for support of broadcast/multicast services. It is presently assumed that the CTCH can be used for all categories of broadcast/multicast services.

For CBS, the CTCH is mapped to a FACH transport channel. The FACH is also a candidate to be used for multicast services in future releases. This possibility will be further investigated in this report.

6 Cell Broadcast Service CBS

This clause contains the requirements derived from GSM and ANSI-41 specifications of Cell Broadcast Service and the analysis of the impact on the radio interface Uu.

The main requirements for Release 99 are:

- service continuity (i.e. no degradation of the GSM and ANSI-41 CBS as seen by users);
- the restrictions regarding the radio interface which are given in GSM or ANSI-41 does not remain any longer;
- the content of this clause should be a basis for future broadcast/multicast service developments;
- minimising the power consumption by use of intelligent scheduling schemes for CB messages. (GSM CB message discontinuous reception (CBS DRX) should become mandatory).

The analysis of 3G CB service (3G-CBS) integration is done top-down. It starts with the network and protocol architecture applicable on each interface.

In subclause 6.1 the impact of CBS (GSM) on UTRAN functions is described. This subclause provides all information on the network level needed to derive the requirements for the radio interface.

In subclause 6.2 the impact of CBS (ANSI-41) on UTRAN functions is described.

Subclause 6.3 discusses the requirements on the radio interface.

Further special radio requirements are listed in subclauses related to each (sub-)layer of the radio interface.

The functions related to the CBC-RNC reference point are not in the scope of RAN WG2.

6.1 CBS (GSM)

6.1.1 Impact on UTRAN functions

6.1.1.1 Network and Protocol Architecture

Figure 6.1 summarises the network and protocol architecture chosen for Release 1999 by S2, T2, R3 and R2. Note that the Cell Broadcast Centre is integrated into the Core Network.

It is aimed to define a radio interface protocol architecture that is independent of the chosen configuration of the CBC-RNC reference point.

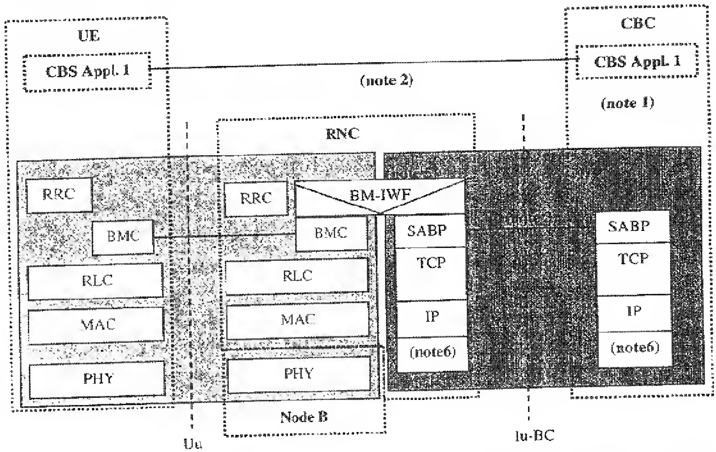

Figure 6.1: General Protocol Architecture when RNS is connected to UMTS Core Network

NOTE 1: S2 has chosen to integrate the CBC into the Core Network

NOTE 2: CBS Application protocol is to be defined by TSG T2

NOTE 3: R3 is responsible for specifying SABP (Service area broadcast protocol, TS 25.419 [19]).

NOTE 4: This relay function provides IP routing to RNC.

NOTE 5: The CBC sends a CB message together with its scheduling information once to an RNC (see 3GPP TS 23.041 and 3G 25.419). The BM Interworking Function (BM-IWF) distributes CB messages received over Appl. 3 to all BMC instances indicated in the delivered cell list. For future releases of UMTS a new function would be necessary if a geographical area is delivered instead of a cell list.

NOTE 6: The lower layer on the Iu-BC interface uses AAL5 over ATM (packet transmission).

In the following, the data unit delivered from/to the CBS Application 1 protocol is denoted as "CB message". This data unit is described in TS 23.041. It comprises the following CB message parameters:

Number-of-Pages (1 octet),
(CBS-Message-Information-Page 1 (82 octets), CBS-Message Information-Length 1(1 octet)) [,
..,
(CBS-Message-Information-Page 15 (82 octets), CBS-Message Information-Length 15)(1 octet)]

This implies a maximum CB message length of 1 + 15 (82+1) = 1246 octets.

NOTE 8: In TS 25.419 [19], the maximum CB message length of 1246 octets is kept by R3.

6.1.1.2 BM-IWF

6.1.1.2.1 Broadcast/Multicast Distribution for GSM based CB messages

The main objective of the BM-IWF in RNC is to distribute the received CB messages towards the BMC entities configured per cell for further processing. This is done in accordance with the associated schedule information.

The radio interface-related schedule information associated with each CB message is described in 3GPP TS 23.041 and is provided in Table 6.1 for information.

Table 6.1: CB Information Elements sent from CBC to RNC for further management

CB Information Element	Description
Message ID	Source and type of CB message
Serial Number	Serial number: Each type of CB message can vary. These variations are expressed by the serial number. The Serial Number consists of three information elements: **Geographical scope** (values: immediate cell wide, PLMN wide, LA wide, cell wide), **Message Code,** **Update Number.**
Data Coding Scheme	Data coding scheme used
Category	Category of the CB message: HIGH: CB message should be broadcast at the earliest opportunity NORMAL: CB message should be broadcast within the associated Repetition Period BACKGROUND: CB message with lowest transmission priority
Repetition Period	Period of time after which broadcast of the CB message should be repeated
Number Of Broadcast Requested	Number of times the CB message is to be broadcast 0: infinitely 1...n: finite number of repetitions

6.1.1.2.2 Broadcast/Multicast Flow Control

When the BMC cannot provide any longer the requested service the BMC is said to be congested. The Broadcast/Multicast Flow Control function takes measures to inform the data source about this congestion situation and to reduce the amount of data to be broadcast or multicast by the congested BMC entity.

6.1.1.2.3 Administrative Data Management

The CBC can request the status of the CBS messages which are currently broadcast. This implies a function that can collect status information which is then reported to the CBC.

6.2 CBS (ANSI-41)

6.2.1 Impact on UTRAN functions

6.2.1.1 Network and Protocol Architecture

Figure 6.2 summarises the network and protocol architecture chosen for Release 1999 by S2, T2, R3, R2 and 3GPP2.

It is aimed to define a radio interface protocol architecture that is independent of the chosen configuration of the CBC-RNC reference point.

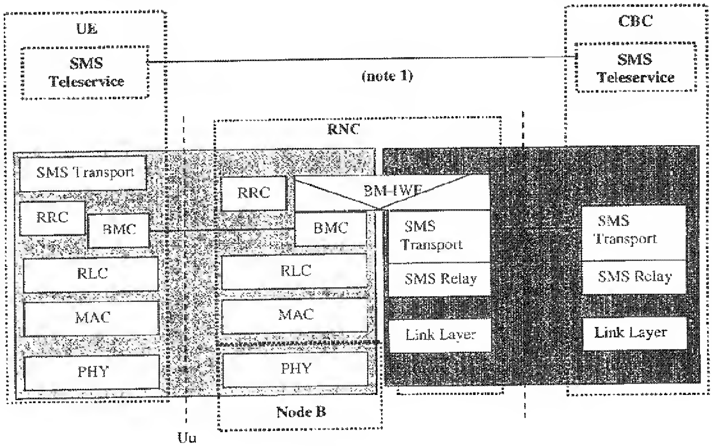

Figure 6.2: General Protocol Architecture when RNS is connected to ANSI-41 Core Network

NOTE 1: This interface is defined in ANSI-41, [20] TIA/EIA-637-A: "TR45 – Short Message Service for Spread Spectrum Systems".

NOTE 2: This interface is defined in [20] TIA/EIA-637-A: "TR45 – Short Message Service for Spread Spectrum Systems".

NOTE 3: The BM Interworking Function (BM-IWF) distributes CB messages received to all BMC instances indicated in the delivered cell information.

NOTE 4: The SMS Relay Layer is defined in [20] TIA/EIA-637-A: "TR45 – Short Message Service for Spread Spectrum Systems". 3GPP2 do not specify any specific Link layer. This could be implementation specific.

In the following, the data unit delivered from/to the SMS Transport layer protocol is denoted as "ANSI-41 CB message". This data unit is described in TIA/EIA-637-A [20].

6.2.1.2 BM-IWF

6.2.1.2.1 Broadcast/Multicast Distribution for ANSI-41 Core Network based CB messages

The main objective of the BM-IWF in RNC is to distribute the received ANSI-41 CB messages towards the BMC entities configured per cell for further processing. This is done in accordance with the associated distribution information.

The ANSI-41 CB messages are described in TIA/EIA-637-A. Each ANSI-41 CB message is to be sent once as soon as possible over the radio interface. Note that no other scheduling is required in the RNC as it is for UMTS based CB messages.

6.3 Radio Interface Requirements

The transmission of CB messages from RNC to UEs via Node B and the cells under its control is in the scope of TSG RAN WG2.

6.3.1 Protocol architecture

Figure 6.2 shows those parts of the radio interface protocol stack which are relevant for CBS. The shown architecture has been selected by RAN WG2.

In the user plane, above the RLC sublayer, a BMC sublayer is introduced (which is assumed as transparent for all services except broadcast/multicast).

At the UTRAN side, the BMC sublayer shall consist of one BMC protocol entity per cell. It is also assumed that each broadcast/multicast requires a single CTCH, which is provided by MAC-c/sh, through the RLC sublayer. The respective RLC entity operates in Unacknowledged mode (UM). This model assumes that there is a function in the RNC above BMC that resolves the geographical area information of the CBC message (or, if applicable, performs evaluation of a cell list) received from the Cell Broadcast Centre (CBC). A BMC protocol entity serves only those messages at BMC-SAP that are to be broadcast into a specified cell.

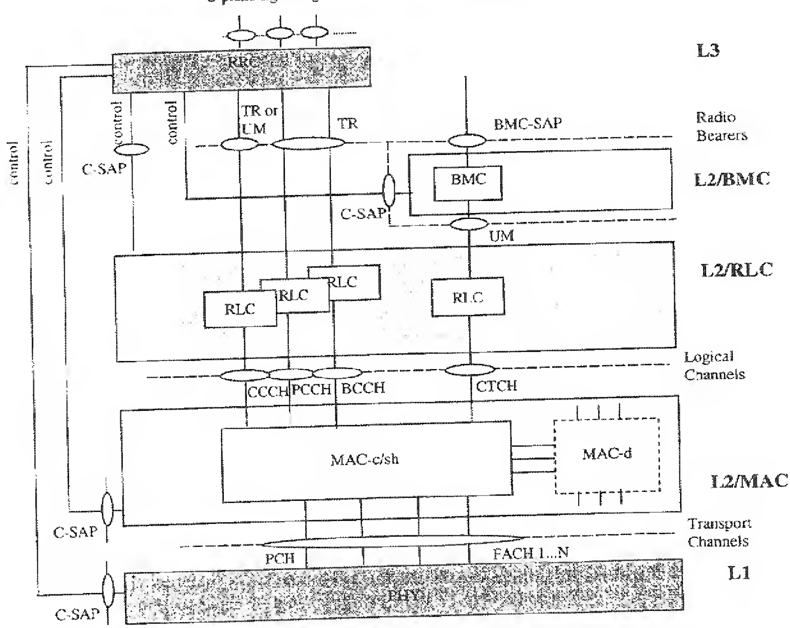

Figure 6.3: Protocol architecture on the radio interface

The following assumptions are made:

- For each cell that supports CBS, one BMC entity should be created in CRNC.
- In the UE one BMC entity should be created when the user has activated CB message IDs.
- It is assumed that one CTCH is created per broadcast/multicast service.

NOTE: For R99 only one CTCH is necessary because only CBS is part of this release.

- The logical channel types BCCH, CCCH, SHCCH (TDD), CTCH, DCCH and DTCH can be mapped onto a FACH.
- A constant TB size is assumed and hence a CTCH should be mapped onto a single FACH.
- CB messages delivered by CBC arrive in BMC as a single packet (BMC SDU (cf. 3GPP TS 23.041).
- For R99 UEs can have the capability to receive CB messages in Idle mode and in Connected mode.
- CB messages are user data delivered in the user plane to BMC.
- Common traffic radio bearer of a cell is established, maintained and released by RRC.
- The RRC (CRNC) configures the CBS related channels via the control SAPs and signals availability of CBS to the peer RRC (UE) via System Information Message which itself configures its lower layers.
- The BMC (CRNC) stores the CB messages arriving over the CBC-RNC-interface and generates the BMC Message sequences.
- Scheduling and DRX procedure:
  There are fixed, periodic allocations for CTCH data on FACH and S-CCPCH. This information is conveyed by RRC (CRNC) via System Information Message on BCCH. The receiving BMC (UE) detects and reads the BMC Schedule Message. Based on its stored schedule information, the BMC (UE) can decide which CB message is new or old. RRC (UE) is informed to instruct the PHY (UE) via C-PHY when it has to listen to the physical channel(s) carrying CBS information.

6.3.2 Examples of procedures

NOTE: The examples provided in the following subclauses are based on the GSM CBS. For ANSI-41 CBS, similar examples can be derived.

Following examples of procedures are described in this subclause:

- CB message storage in BMC entity and counting of CB message repetition (subclause 6.2.2.1);
- BMC message scheduling (including CBS related radio resource configuration and system information broadcast) (subclause 6.2.2.2);
- Activation of CB message reception in the UE by the User (subclause 6.2.2.3);
- CB message reception with CB DRX (subclause 6.2.2.4);
- BMC Overflow (subclause 6.2.2.5);
- BMC Underflow (subclause 6.2.2.6).

6.3.2.1 CB message storage in BMC entity and counting of CB message repetition

Precondition:
A BMC entity on the network side serving a specific cell is created by O&M when CBS support is activated for this cell.

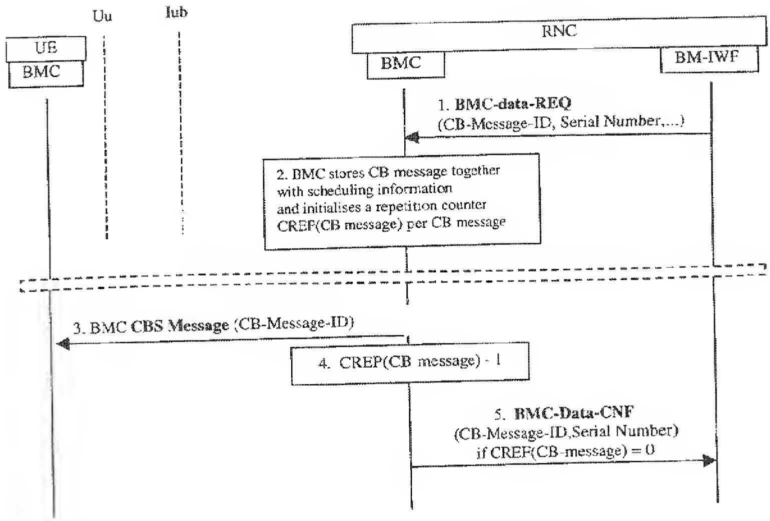

Figure 6.4: Example of Message Flow for broadcast of CBS related system information

The example message sequence for broadcast of CBS related system information is described as follows (numbering refers to message numbering in the figure, parameters given in square brackets are optional):

1. BMC receives the CB message together with scheduling information from CBC:
   BMC-Data-REQ (CB message, Schedule information) with

   CB message:
   Message-ID,
   [Old-Serial-Number],
   New-Serial-Number,
   Data-Coding-Scheme,
   CB-Data

   Scheduling information:
   [Category],
   Repetition-Period,
   Number-of-Broadcasts-Requested

NOTE: For R99 the CB-Data equals to the term CB message which is introduced in subclause 6.1.1.

The description of the listed parameters is given in TS 23.041 [13].

2. BMC stores the CB message and the Scheduling information.
   This is necessary because it is only received once but have to be transmitted n times over the radio interface, where n = Number-of-Broadcasts-Requested.

   For each CB message a repetition counter CREP is created if Number of Broadcast Requested is finite.

   A CB message is completely identified by the pair (Message-ID, Serial-Number).

   If the primitive do not contain the Old-Serial-Number parameter it should be the first time this CB message is delivered. If this is not the case, an error indication BMC-Error-IND(Cause=Message ID already stored) should be given to the BM-IWF.

   If the primitive contains the Old-Serial-Number parameter an existing CB-Message should be replaced. If the indicated old CB message is not stored an error indication BMC-Error-IND(Cause=old CB message not stored) should be given but the delivered CB message should be stored and handled as a new CB message.

Table 6.2: Mapping between Primitive parameters and BMC PDU information elements

Parameter of BMC-Data-REQ (TS 25.324 [16]))	Information element of BMC CBS Message (TS 25.324 [16]))	Parameter of BMC-Data-IND (TS 25.324 [16]))
Message-ID	Message ID	Message-ID
[Old-Serial-Number]	---	not applicable
New-Serial-Number	Serial Number	Serial Number
Data-Coding-Scheme	Data Coding Scheme	Data Coding Scheme
CB-Data	CB Data	CB-Data

3-5. The CREP(CB-Message-ID) is decreased each time this CB message is broadcast. When CREP(Message-ID) equals 0 an indication BMC-Data-CNF(Message-ID) is given to BM-IWF that the task is finished

6.3.2.2 BMC message scheduling

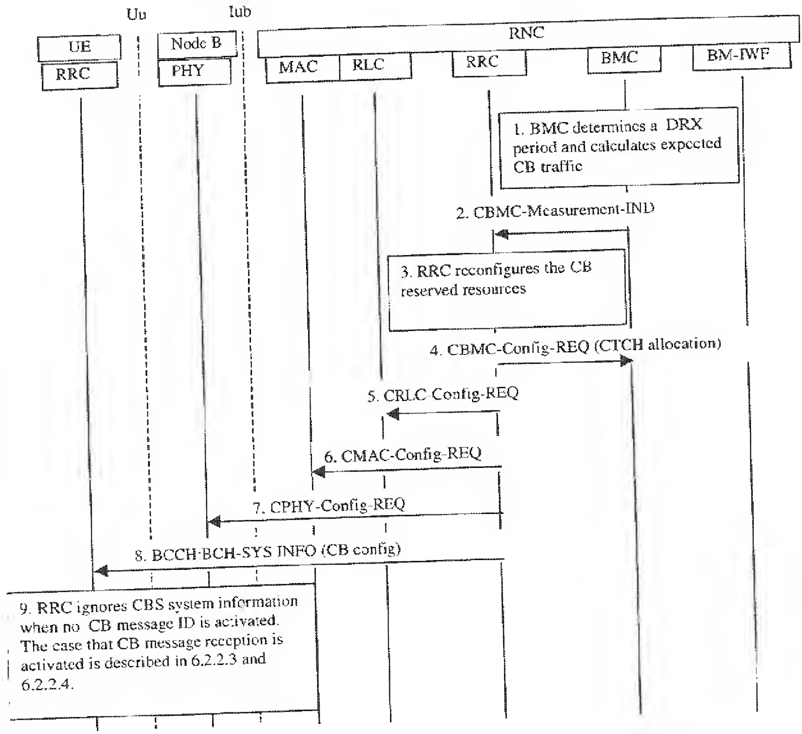

Figure 6.5: Example of Message Flow for BMC message scheduling

The example sequence for BMC message scheduling is described as follows (numbering refers to message numbering in the figure):

1. BMC calculates periodically a CBS DRX period. If CBC has assigned a DRX period to the message this information shall be taken into account by the scheduling scheme.
   BMC schedules continuously the CB message sequence which has to be transmitted during the current and next scheduling period. A result of the schedule calculations performed in BMC is the overall CB traffic volume.

   When BMC is requested to send a CB message, the following scenarios may occur:

   - Case 1: It is the first time that a CB message is sent in a specific cell.
   - Case 2: Transmission of other CB messages is already activated.
   - Subcase 2.1: Within the current CBS DRX period reserved radio resources which are marked as "new" could be used to sent the CB message immediately.
   - Subcase 2.2: CB message could be sent the first time in the next CBS DRX period and the reserved CB capacity is enough.
   - Subcase 2.3: CB message could be sent the first time in the next CBS DRX period, but the currently reserved radio resource capacity is too low.

2, 3. For case 1 and subcase 2.3 the BMC indicates to RRC the CB traffic volume using the primitive CBMC-Measurement-IND. RRC checks whether more radio resources can be reserved for CTCH traffic. If possible, RRC reconfigures RLC, MAC and PHY at the network side and informs the peer RRC entities (see step 8). If not possible, the configuration remains as it is.

4. RRC informs BMC about the successful/unsuccessful reconfiguration (acknowledgement on 2.) with the primitive CBMC-Config-REQ (CTCH configuration):

   - If RRC could not provide enough CB resources flow control mechanism should be initiated by BMC; primitive BMC-Congestion-IND is used to indicate to BM-IWF the congestion situation. For flow control see subclauses 6.3.2.5 and 6.3.2.6.

5. RRC configures RLC (if necessary).

6. RRC configures MAC (if necessary).

7. RRC configures PHY (in Node B) (if necessary).

8. The reconfiguration of CBS resource allocation is broadcast by SYSTEM INFORMATION message to the UE. The CBS related system information is carried by BCCH mapped to BCH. 9. If CB message reception is not activated by the UE, RRC ignores this system information.
   Otherwise the RRC configures its lower layers regarding the received configuration information. For details see 6.3.2.2 and 6.3.2.3.

6.3.2.3 Activation of CB message reception by User

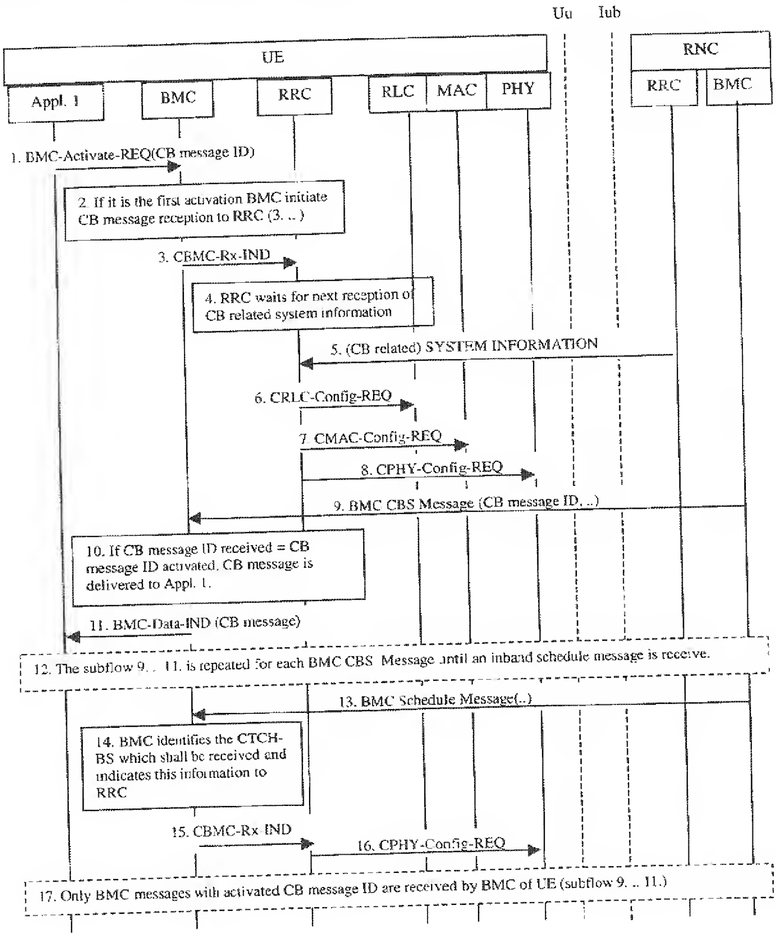

Figure 6.6: Example of Message Flow for activation of CB message reception by User

The example sequence for activation of CB message reception by the user is described as follows (numbering refers to message numbering in the figure):

1. The user activates a Message ID by the primitive BMC-Activation-REQ.

   (It is assumed that a BMC entity is already established).

NOTE: This primitive is not yet defined by T WG 2 SWG 3.

2, 3. If it is the first time the user activates a Message ID the BMC shall indicate to RRC that CB message reception shall start. This is done by primitive CBMC-Rx-IND.

4, 5. When RRC receives such an indication first it has to wait for the next CB related SYSTEM INFORMATION message to read the general CB configuration.

6...8. The RRC configures the CBS related radio resources.

9...13. All BMC messages are read by BMC until a first BMC Schedule message is received. A BMC CBS Message consists of the IEs Message ID, Serial Number, Data Coding System and CB Data. The received CB messages are only delivered by the primitive BMC-Data-IND to the CB application if the received Message ID and/or the received Serial Number are new. The BMC messages are described in TS 25.324 [16].

14, 15. The BMC Schedule message informs which CB messages will be sent when in the next DRX schedule period. The BMC indicates to the RRC only those BMC messages which should be received by the UE (CBMC-Rx-IND). The decision which BMC messages are of interest is based on the activated Message IDs and the comparison of the Serial Number received with which that is currently stored in the UE.

16. The RRC configures the PHY layer at which time intervals it should receive on the CBS related radio resources.

17. As an consequence only CB messages of interest are received by the BMC of the UE and delivered to the CB application.

6.3.2.4 CB message reception with DRX

Precondition:

Under normal condition BMC Schedule messages are received each time when they are sent. When a BMC Schedule Message is corrupted the BMC should read all BMC messages continuously until a new BMC Schedule Message is found.